CSRD-83-02

Permanent Counting Stations – Master Software

W.J. Wiercieński
Project Research Engineer
Control and Information Systems

Published by:
The Transportation Technology and Energy Branch
Ontario Ministry of Transportation and Communications
Hon. James W. Snow, Minister
H.F. Gilbert, Deputy Minister

Published without prejudice
as to the application of the findings.
Crown copyright reserved; however, this
document may be reproduced for non-commercial
purposes with attribution to the Ministry.

For additional copies write:
The Editor
Research and Development Branch
Ministry of Transportation and Communications
1201 Wilson Avenue
Downsview, Ontario
M3M 1J8
January 1983

Abstract

Large sums of money are spent yearly for the maintenance and development of highways in Ontario. To assist in deciding the most effective means of pursuing such maintenance and development, permanent counting stations have been established to provide information regarding vehicular use of the highways.

It has been proposed that these stations be replaced with microprocessor-based stations which are controlled by software programs. There are both system programs (which control the operation of the system) and application programs (which collect and calculate data). It is the purpose of this document to describe in detail the system software. The application software is discussed here only superficially. For a detailed description of the application software refer to "Permanent Counting Stations -- Microcomputer Programming Manual" (LAB-82-03).

TABLE OF CONTENTS

PAGE

1/ GENERAL DESCRIPTION 1

2/ CENTRAL STATION HARDWARE 7

3/ FIELD STATION DESCRIPTION 9
3.1/ General 9
3.2/ Microprocessor. 9
3.3/ Modem 11
3.4/ Definition of Roadway 11
3.5/ Detector Loops. 11
    3.5.1/ Physical Characteristics 11
    3.5.2/ Loop Wiring. 14
    3.5.3/ Practical Considerations for Loop Installation 14
3.6/ Lightning Protection. 14

4/ DEFINITION OF DATA TO BE COLLECTED 19
4.1/ Volume Counts 19
4.2/ Speed Classification. 20
4.3/ Length Counts 20

5/ KEYBOARD OPERATIONS 21
5.1/ Overview 21
5.2/ Display Modes 21
    5.2.1/ Blank Mode 21
    5.2.2/ Time Mode 21
    5.2.3/ Hex Display Mode 22
    5.2.4/ Hex/Binary Display Mode. 22
    5.2.5/ Power-Up Display 22
    5.2.6/ Other Displays 23
5.3/ Different Displays and the Meaning of the Keys 23
5.4/ Examining Memory 23
5.5/ Changing Memory 24
    5.5.1/ Entering New Hex Data. 25
    5.5.2/ Changing Bits. 26

PAGE

5.5.3/ Clearing Memory 26
5.5.4/ Protected Locations 27
5.6/ Stepping Through Memory 27

6/ CONTROLLER INITIALIZATION 41
6.1/ Set Security Password 41
6.2/ Set Station Address 41
6.3/ Set Date and Time 42
6.4/ Define Station Geometry 42
6.5/ Define Volume Counts 46
6.6/ Detector Spacing and Width. 47
6.7/ Master Initialization 50

7/ PROGRAM DESCRIPTIONS 51
7.1/ Overview of Program Structure 51
7.1.1/ Routines Not Driven by Interrupts. 51
7.1.2/ Routines Driven by Interrupts. 56
7.2/ Power-Up Routine (POWRON) 57
7.3/ Executive Routine (EXEC) 67
7.4/ Clock Routine (CLOCK) 72
7.4.1/ Find the Number of Days in a Given Month Subroutine (NOFDAY). 74
7.4.2/ Daylight Savings Adjustment Subroutine (DAYLIT). . . . 75
7.5/ Keyboard Routine (KEYBD). 88
7.5.1/ Display for State 0 Subroutine (DISPL0). 93
7.5.2/ Display for State 1 Subroutine (DISPL1). 93
7.5.3/ Display for State 2 Subroutine (DISPL2). 93
7.5.4/ Display for States 3 and 6 Subroutine (DISPL3, DISPL6) 93
7.5.5/ Display for States 4 and 7 Subroutine (DISPL4, DISPL7) 93
7.5.6/ Display for States 5 and 8 Subroutine (DISPL5, DISPL8) 94
7.5.7/ Display for State 9 Subroutine (DISPL9). 94
7.5.8/ Key Pushed While in States 0 or 9 Subroutine (PUSH0, PUSH9) 94
7.5.9/ Key Pushed While in State 1 Subroutine (PUSH1) 95

PAGE

- 7.5.10/ Key Pushed While in State 2 or 3 Subroutine (PUSH2, PUSH3) 95
- 7.5.11/ Key Pushed While in State 4 Subroutine (PUSH4) 95
- 7.5.12/ Key Pushed While in State 5 Subroutine (PUSH5) 95
- 7.5.13/ Key Pushed While in State 6 Subroutine (PUSH6) 96
- 7.5.14/ Key Pushed While in State 7 Subroutine (PUSH7) 96
- 7.5.15/ Key Pushed While in State 8 Subroutine (PUSH8) 96
- 7.5.16/ Process Address Change Subroutine (ADDCHG) 97
- 7.5.17/ Determine If Address Can Be Read Subroutine (READOK) 98
- 7.5.18/ Determine If Memory Can Be Changed Subroutine (WRITOK) 98

7.6/ Communication Routines 124
  - 7.6.1/ Main Routine (COMM) 124
  - 7.6.2/ Receiving Routine (RECV) 128
    - 7.6.2.1/ Prepare for End of Message after Failure Subroutine (EOMAF) 143
  - 7.6.3/ Transmit First Character Routine (TFIRST) 145
  - 7.6.4/ Transmit Routine (XMIT) 148
  - 7.6.5/ Prepare Answer Routine (PREANS) 160
    - 7.6.5.1/ Prepare Logical Error Message Subroutine (LOGER) 164
    - 7.6.5.2/ Reset Control Subroutine (RESCON) 165
  - 7.6.6/ Communications Timer Routine (TIMER1) 181
  - 7.6.7/ Security Code Timer Routine (TIMER2) 184

7.7/ Master Initialization Routine (MASINT) 186

7.8/ Initiate PROM Checksum Routine (CHECK24) 189
  - 7.8.1/ Perform Checksum (1 Byte) Subroutine (CHECK60) 191
  - 7.8.2/ Compute PROM Checksum Subroutine (CALPRO) 193

7.9/ RAM Test Routines 198
  - 7.9.1/ Test 1 Byte of RAM Routine (RAM60) 198
  - 7.9.2/ Perform RAM Test Routine (RAMCHK) 200

7.10/ Power-Down Routine (IGNORE) 203

APPENDIX A/ Communications Messages 205

APPENDIX B/ Setting Up of the PROM Checksum Tables 221

APPENDIX C/ Program Listings 225

1/ GENERAL DESCRIPTION

The Ontario Ministry of Transportation and Communications (MTC) is responsible for maintaining the King's Highways system in Ontario. A large amount of money is spent for the maintenance of old highways and the development of new ones. In order to spend this money in an intelligent manner, approximately 20 permanent counting stations (PCS) have been established throughout Ontario to provide data on the number of vehicles passing a station in a certain period of time. These data, supplemented occasionally by manual volume counts, provide a basis for making intelligent and effective decisions regarding highway maintenance and planning.

The present PCS system includes approximately 20 stations, as listed in Table 1. Each station is enclosed in a plastic hut located near the road for which the data are being collected. Detectors imbedded in the road are connected to electro-mechanical counters which record volume data with a paper punch. Every three or four weeks, the resulting volume counts punched on paper tapes are collected and taken to a central location (thereby incurring labour and transportation costs) where the tapes are processed by a tape reader/translator, and the data are then analysed by a computer program. The whole process of manually gathering the tapes and having them processed is labour-intensive and time-consuming. Furthermore, the PCS is largely mechanical and, consequently, it is unreliable and subject to failure. These failures cannot be detected until the station is visited by field personnel, which could be an interval as long as one month. Sometimes an equipment failure may only become apparent after the data have been collected, transported and analysed. The equipment has thus become costly to maintain. In addition, the current system is inflexible, providing nothing but volume counts. These shortcomings have prompted replacement of the obsolete PCS equipment with a microprocessor-based automatic system.

Table 1/ Permanent Counting Stations

TELEMETRY SYSTEM PERMANENT COUNTING
STATIONS PROPOSED FOR IMPLEMENTATION

EXISTING TEST STATIONS	ADDITIONAL STATIONS	PROPOSED FUTURE STATIONS
DIXIE	BELFIELD	HWY 403 AT QEW
PORT SEVERN	BRADFORD	BURLINGTON SKYWAY
	BRAMPTON	LONDON (HWY 401)
	DUNVEGAN	HWY 7 (WEST OF PERTH)
	GRAVENHURST	HWY 11 (N. OF N. BAY)
	HOMER	HWY 11/17 (W. OF T. BAY)
	KEELE	
	KIRBY	
	LIVERPOOL	
	MAPLE	
	NIPIGON	
	OTTAWA	
	PORT HOPE	
	ROWENA	
	SHELBURNE	
	SIMCOE	
	SNELGROVE	

The new PCS system utilizes a standard model 170 microprocessor-based traffic signal controller (already used extensively by MTC for other applications). At each station, all the mechanical counters/punches are to be replaced by a single 170 controller. The old stations will still use the plastic huts, but new stations will use a standard traffic cabinet. Each station will have a communication link (via modem and public telephone line) to a central computer in Toronto. Note that this computer currently has spare capacity, so implementation of the new PCS system will not have any impact on the computer's performance. The central computer now has an automatic dial-up unit and a modem. Thus, it will be able to poll each PCS over the public telephone line and collect data. See Figure 1 for a diagram of the setup. Because of the distances between PCS stations, a private dedicated line would be prohibitively expensive (refer to Figure 2 for the locations of the PCS stations, existing and proposed). This new PCS system overcomes the disadvantages of the current system. Some of the more important features of the new system follow.

- Immediate access to data via phone line. Data can be collected daily (or as often as needed) instead of monthly.
- No need for personal visits to stations.
- Equipment is reliable (the controller's mean time between failures exceeds one year).
- Most equipment failures can be automatically diagnosed and reported the next time the central computer calls the station.
- Data collection is flexible. Not only volume data, but speed and length data are available.
- The configuration of lanes for collection of data is flexible and can be changed remotely by the central computer.

- Initializing and resetting of the system can also be done remotely.

In summary, the new PCS system provides a data collection method of greater flexibility and reliability. In addition, it requires less maintenance, resulting in a less labour-intensive system.

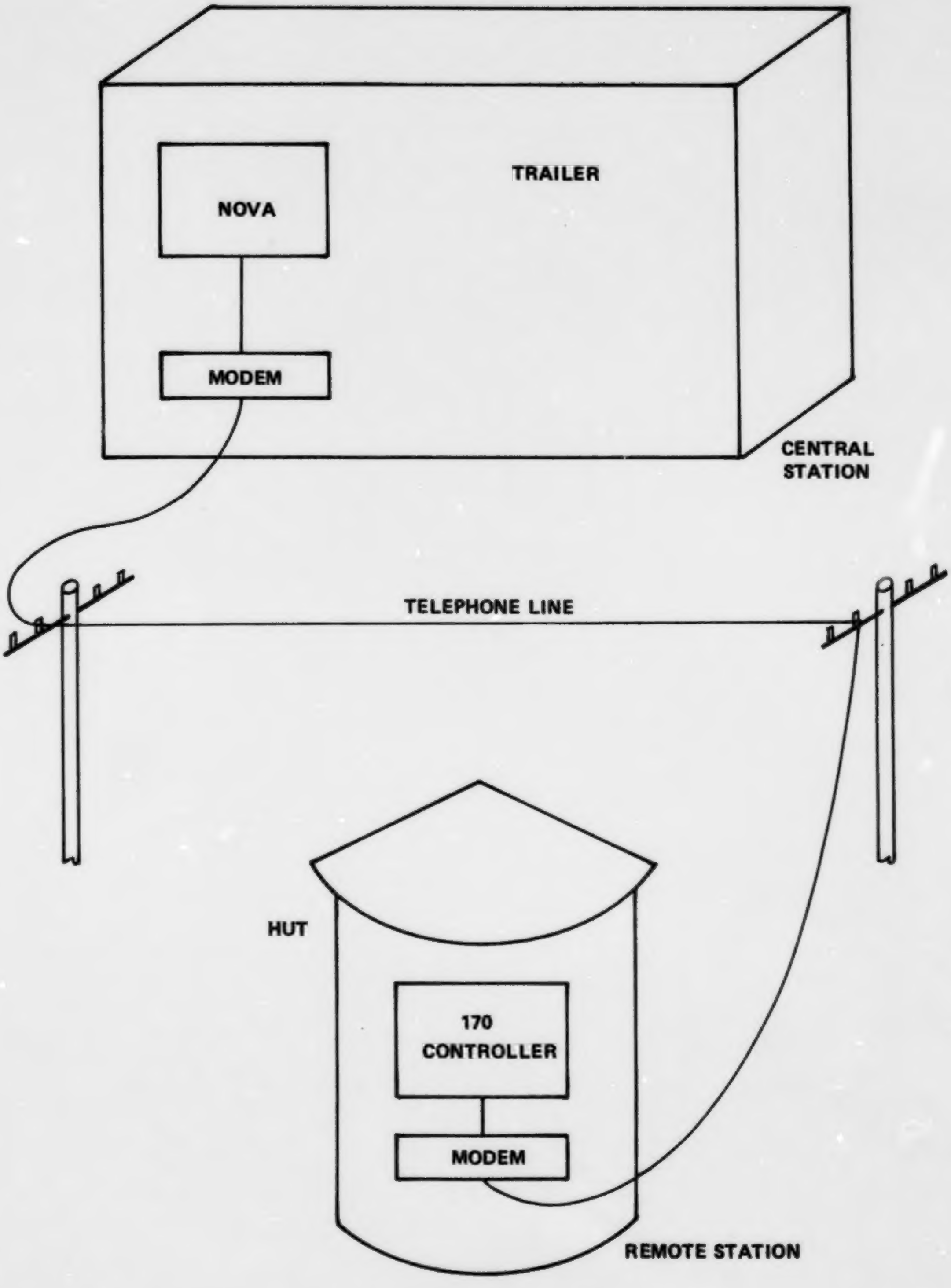

TRAILER
NOVA
MODEM
CENTRAL STATION
TELEPHONE LINE
HUT
170 CONTROLLER
MODEM
REMOTE STATION

Figure 1/ PCS Remote Station and Central Station

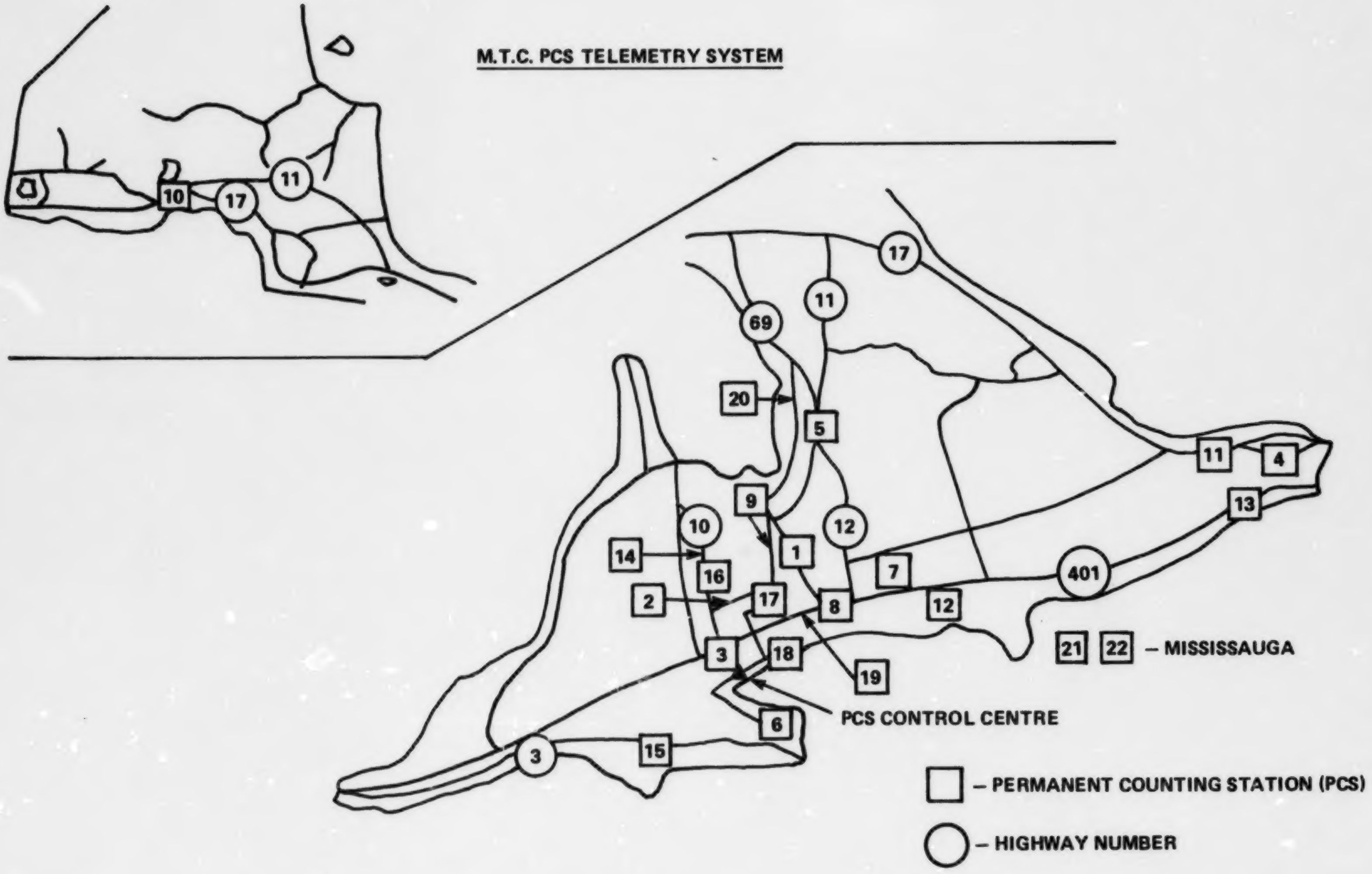

M.T.C. PCS TELEMETRY SYSTEM
10
17
11
17
11
69
20
5
9
10
12
14
1
16
7
401
11
4
13
2
17
8
12
3
18
19
21
22
– MISSISSAUGA
PCS CONTROL CENTRE
6
3
15
– PERMANENT COUNTING STATION (PCS)
– HIGHWAY NUMBER

Figure 2/ Current PCS Locations

2/ CENTRAL STATION HARDWARE

The central computer is a Data General Nova 4/X minicomputer housed in a small trailer near Highway 10 and the QEW in Mississauga. This computer was originally installed for the QEW ramp metering system. Its spare capacity permits use of the computer by the PCS telemetry system for a nominal daily operating time of about 30 minutes. See Figure 3 for a diagram of the central computer and its input/output devices.

The NOVA has 256K bytes of memory, a real-time clock, and auto program load. It uses a 10 megabyte disk unit which can store up to one month of PCS data, and a 9-track magnetic tape unit which can be used to transfer PCS data to another computer system. A system keyboard console allows the operator to control the PCS system operation. A colour graphic CRT terminal displays the operation status with a labelled map of Ontario, highlighting a PCS station whenever it is being polled by the NOVA (see Figure 2 for this map). The computer is connected via an RS232C interface to the telephone line equipment, consisting of a Bell 801C ACU (one-line automatic dialing unit) to call each PCS remote station automatically and a Bell 103A2S data modem to transfer data to/from the remote station.

The PCS software is run by the NOVA standard operating system. Basically, the operator can use this software to perform the following:

- transfer volume data from a remote PCS station;
- transfer speed and length data from a remote PCS station;
- initialize a station;
- obtain a memory dump of the controller; and
- change the controller's memory.

Polling for data can be started manually by the operator or automatically (for weekends, holidays, et cetera). The volume, speed and length data are eventually transferred to magnetic tape for further processing on an IBM mainframe computer at the MTC Downsview complex.

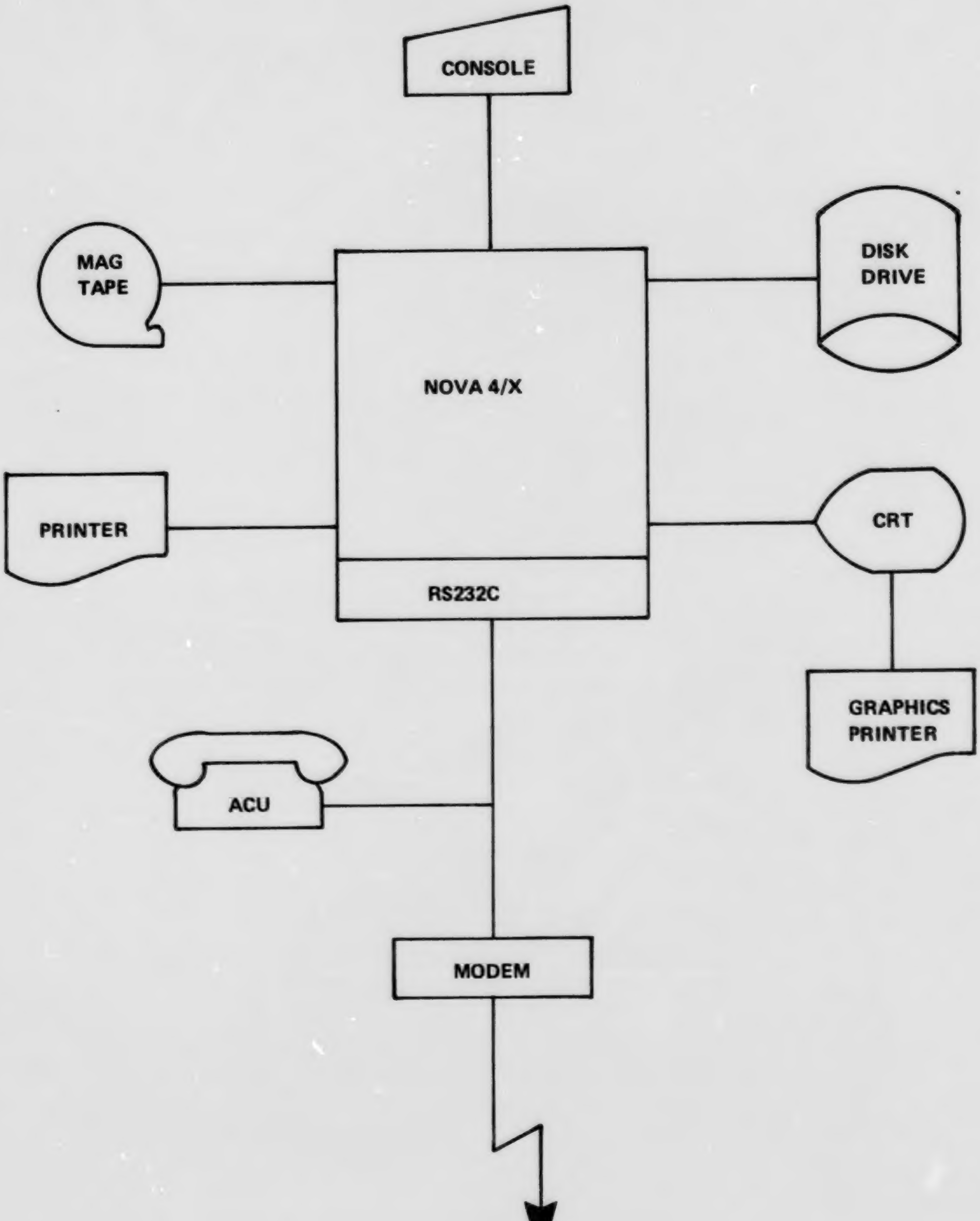

CONSOLE
MAG TAPE
DISK DRIVE
NOVA 4/X
PRINTER
CRT
RS232C
GRAPHICS PRINTER
ACU
MODEM
TO TELEPHONE LINE

Figure 3/ Central Station Hardware

3/ FIELD STATION DESCRIPTION

3.1/ General

Each field station houses a model 170 microprocessor, a modem, detector amplifiers and a phone. Under the pavement lie the detector loops, one per lane if only volume counts are desired, or two per lane for volume, speed and length counts. The old field stations are housed in plastic huts, whereas the new ones will be housed in traffic cabinets. (Refer to Figure 4 for a diagram of a PCS remote station.)

3.2/ Microprocessor

The model 170 microprocessor, used by the province as a standard traffic computer, is built on the Motorola 6800 microprocessor chip. The Topping model 170 controller, used for this application, has a RAM memory capacity of up to 20K bytes (in 2K increments) and 16K or 32K bytes of PROM (a standard controller usually has 1K byte of RAM and 8K bytes of PROM). The programs are written on EPROMs (Erasable Programmable Read Only Memory), so they can be easily modified as required. There are 44 inputs and 52 outputs available to the user, allowing great flexibility. A 1/60 second clock interrupt is used by the software to keep the correct time, and for power-downs of less than 255 minutes, a crystal clock (or downtime accumulator -- DTA) keeps track of the time. During power-downs, the RAM is preserved by batteries for up to 30 days.

The highway detectors are connected directly to the input ports on the microprocessor controller via detector amplifiers. The controller is equipped with an Asynchronous Communication Interface Adapter (ACIA) and an RS232C interface which connects to the data modem. The modem is connected to the public telephone line, allowing communication with the central computer.

The working temperature range for this controller is -37°C to 74°C.
For more details, refer to:

Model 170 Microprocessor Controller Instruction Manual,
Topping Electronics Limited,

and Traffic Signal Control Equipment Specifications,
Highway Engineering Division, Traffic Engineering Office, MTC.

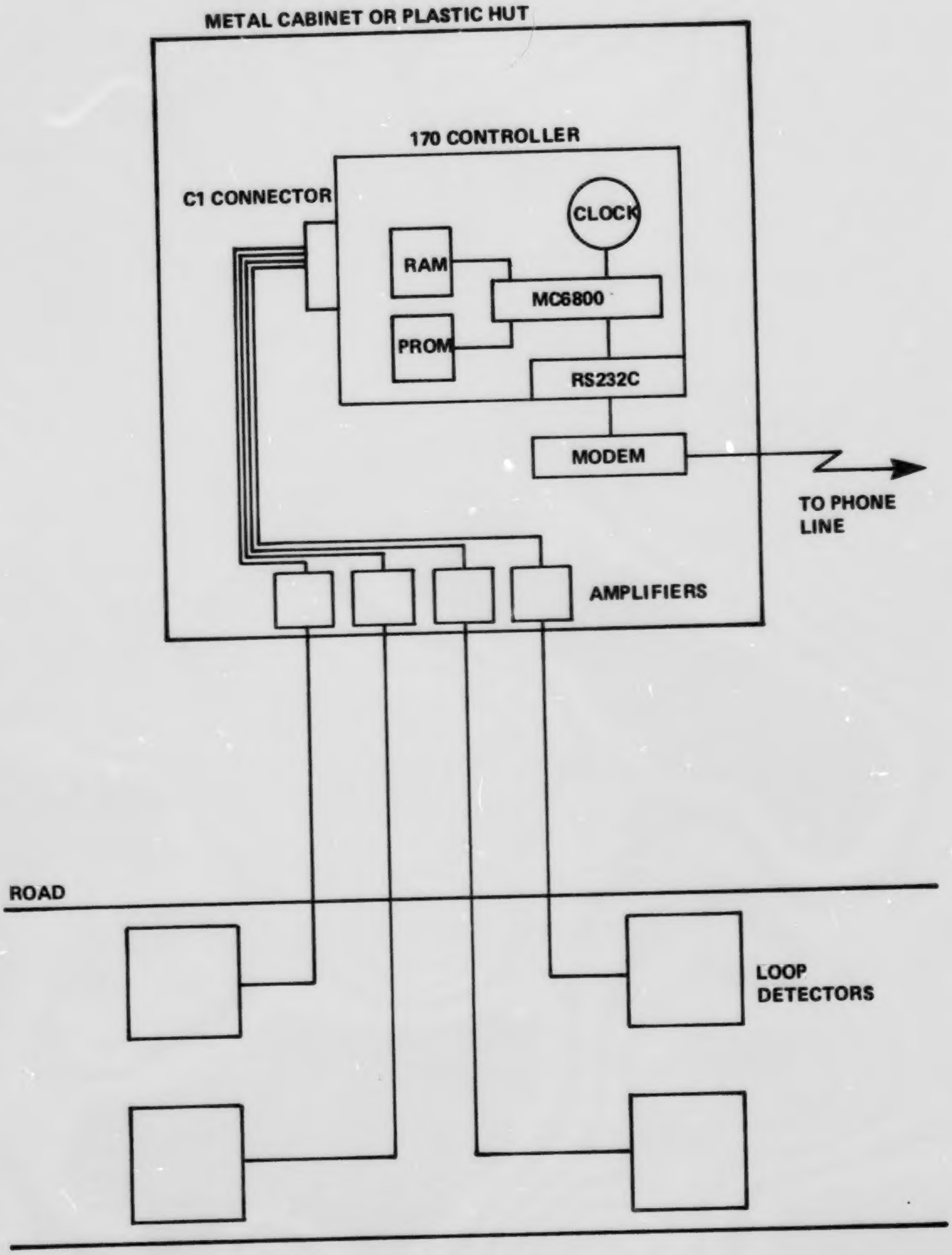

METAL CABINET OR PLASTIC HUT
170 CONTROLLER
C1 CONNECTOR
CLOCK
RAM
MC6800
PROM
RS232C
MODEM
TO PHONE LINE
AMPLIFIERS
ROAD
LOOP DETECTORS

Figure 4/ PCS Field Station Hardware (Number of Loop Detectors and Amplifiers Varies with Station)

3.3/ Modem

The modem employed in the field station is a standard Bell 103A 300 baud modem set in auto-answer mode. The operating temperature is 0°C to 50°C, making it difficult to use in the winter. However, the model 170 controller is designed to house a model 400 modem, which, if equipped with auto-answer capability, would be more suited to the job. Specifications for the Modem 400 can be found in Traffic Signal and Control Equipment Specifications.

3.4/ Definition of Roadway

A road is divided into two directions, Direction 1 and Direction 2. For the sake of establishing a standard, Direction 1 is defined as the direction of north- or westbound traffic; Direction 2, as the south- or eastbound traffic. These directions are divided by some sort of median, be it a line painted on the road or a physical barrier.

Each lane is numbered from 1 to a maximum of 8 starting at the lanes closest to the median (see Figure 5). Thus, any lane is uniquely defined by its direction (1 or 2) and its lane number (1-8).

3.5/ Detector Loops

3.5.1/ Physical Characteristics

The detector loops are current-carrying cables buried beneath the pavement. When a vehicle passes over a loop, a change in the circuit inductance is detected. A lane requires one loop if only volume data are to be collected, whereas two loops are needed if speed and length data are also required. For lanes with two loops, the loop a vehicle first passes over is called the "upstream loop", and the other loop is called the "downstream loop" (see Figure 6). For lanes with only one loop, the loop is considered to be an upstream loop.

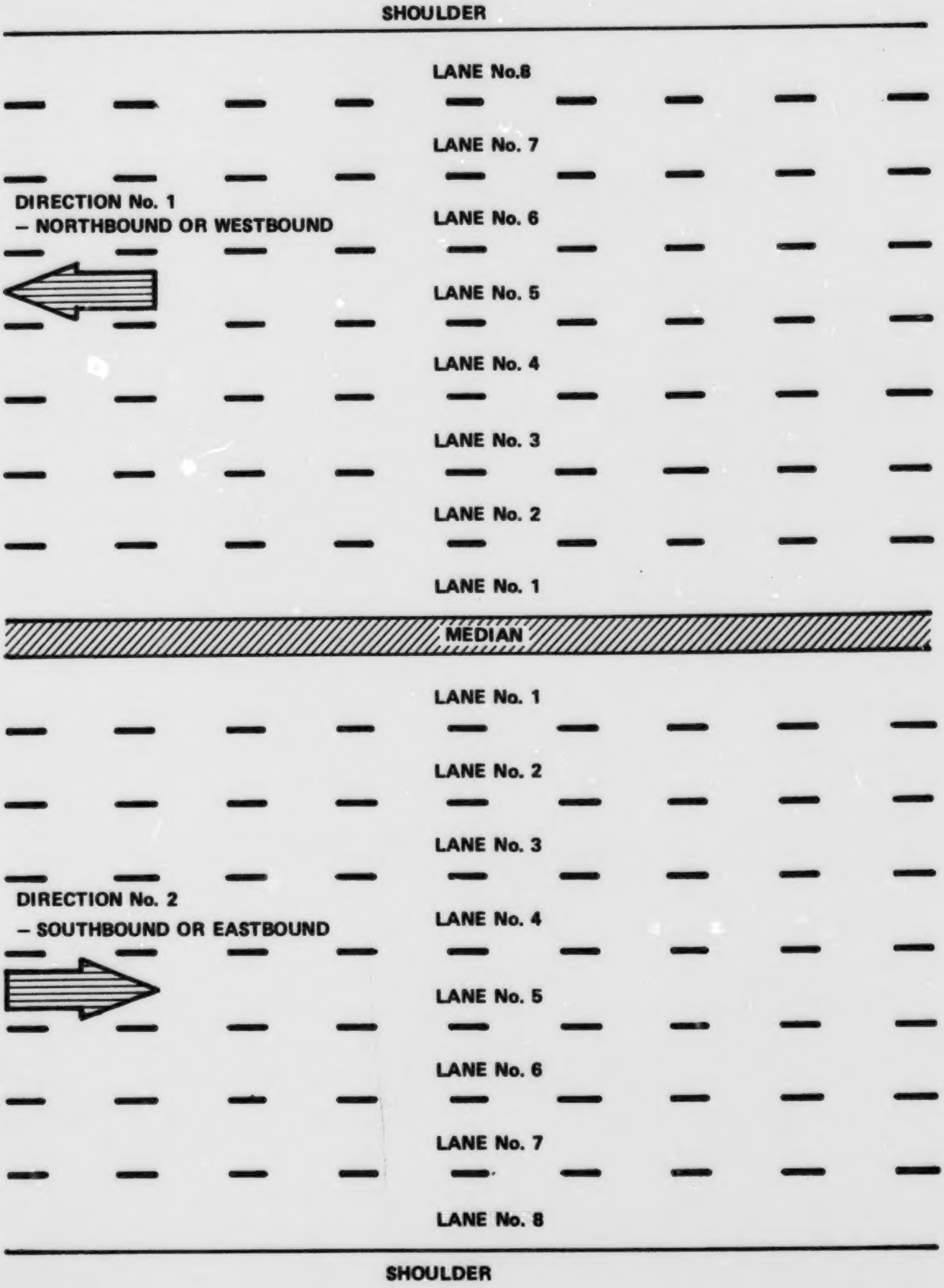

SHOULDER
LANE No.8
LANE No. 7
DIRECTION No. 1
– NORTHBOUND OR WESTBOUND
LANE No. 6
LANE No. 5
LANE No. 4
LANE No. 3
LANE No. 2
LANE No. 1
MEDIAN
LANE No. 1
LANE No. 2
LANE No. 3
DIRECTION No. 2
– SOUTHBOUND OR EASTBOUND
LANE No. 4
LANE No. 5
LANE No. 6
LANE No. 7
LANE No. 8
SHOULDER

Figure 5/ Definition of Roadway

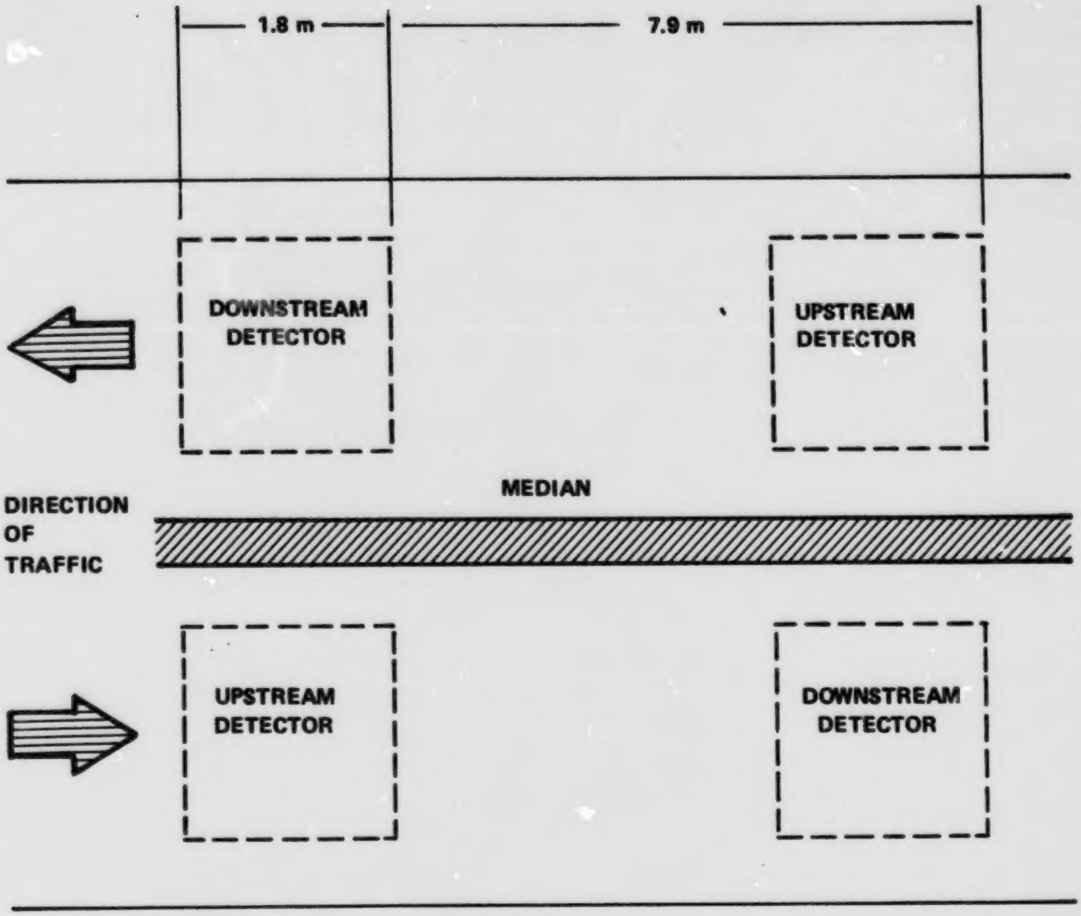

1.8 m
7.9 m
DOWNSTREAM DETECTOR
UPSTREAM DETECTOR
MEDIAN
DIRECTION OF TRAFFIC
UPSTREAM DETECTOR
DOWNSTREAM DETECTOR

Figure 6/ Detector Loops

3.5.2/ Loop Wiring

Each detector loop is connected through an amplifier to a pin on the C1 connector on the 170 traffic controller. The software can access these detectors by reading port locations $5001, $5002, $5003 and $5006. Each detector has an associated bit at one of these locations. The bit will be set if the detector senses the presence of a vehicle and reset otherwise. In each input port, bit 0 corresponds to Lane 1, bit 1 to Lane 2, and so on (see Figure 7 for detector inputs and C1 connector pin assignments). Refer to Figure 8 for an example showing how a 6-lane highway should have its detectors connected. Note, if there is only one detector loop in a lane, it is wired as an upstream detector.

3.5.3/ Practical Considerations for Loop Installation

Based on observations and analysis of traffic data, there are certain suggestions that should be made regarding the installation of detector loops. They are all based on the principle that the loop cables in one lane should be as far away from cables in another lane as possible.

To minimize interference from vehicles travelling in adjacent lanes, the loops should be spread apart as far as is reasonable. Detectors in the edge lanes should be offset toward the edge and those in central lanes should be centered (see Figure 9(a)). To reduce the amount of cable near the edge of a lane, the shape of the loop could be changed to a rectangular or diamond shape (see Figure 9(b)). Also, as shown in Figure 8, detectors in adjacent lanes should be staggered. The feeding cables should be at least 60 cm apart.

3.6/ Lightning Protection

It is important to ensure the PCS field equipment and its power supply both have their own lightning protection. Each such grounding system should consist of three to five 10-foot long lightning rods as established by a ground resistance test. This should be supplemented by electronic circuit protection.

The details can be found in Directive B-130 entitled, "Lightning Protection/Surge Arrestors for Traffic Signal Control Systems" and in the excellent article written by P. Thomas entitled, "Protection for Electrical and Electronic Equipment from the Effects of Lightning."

UPSTREAM DETECTORS, DIRECTION No. 1

PORT 1 ($5001)

PIN No.	46	45	44	43	42	41	40	39
LANE No. (OR BIT No.)	8	7	6	5	4	3	2	1

DOWNSTREAM DETECTORS, DIRECTION No. 1

PORT 2 ($5002)

PIN No.	54	53	52	51	50	49	48	47
LANE No. (OR BIT No.)	8	7	6	5	4	3	2	1

UPSTREAM DETECTORS, DIRECTION No. 2

PORT 3 ($5003)

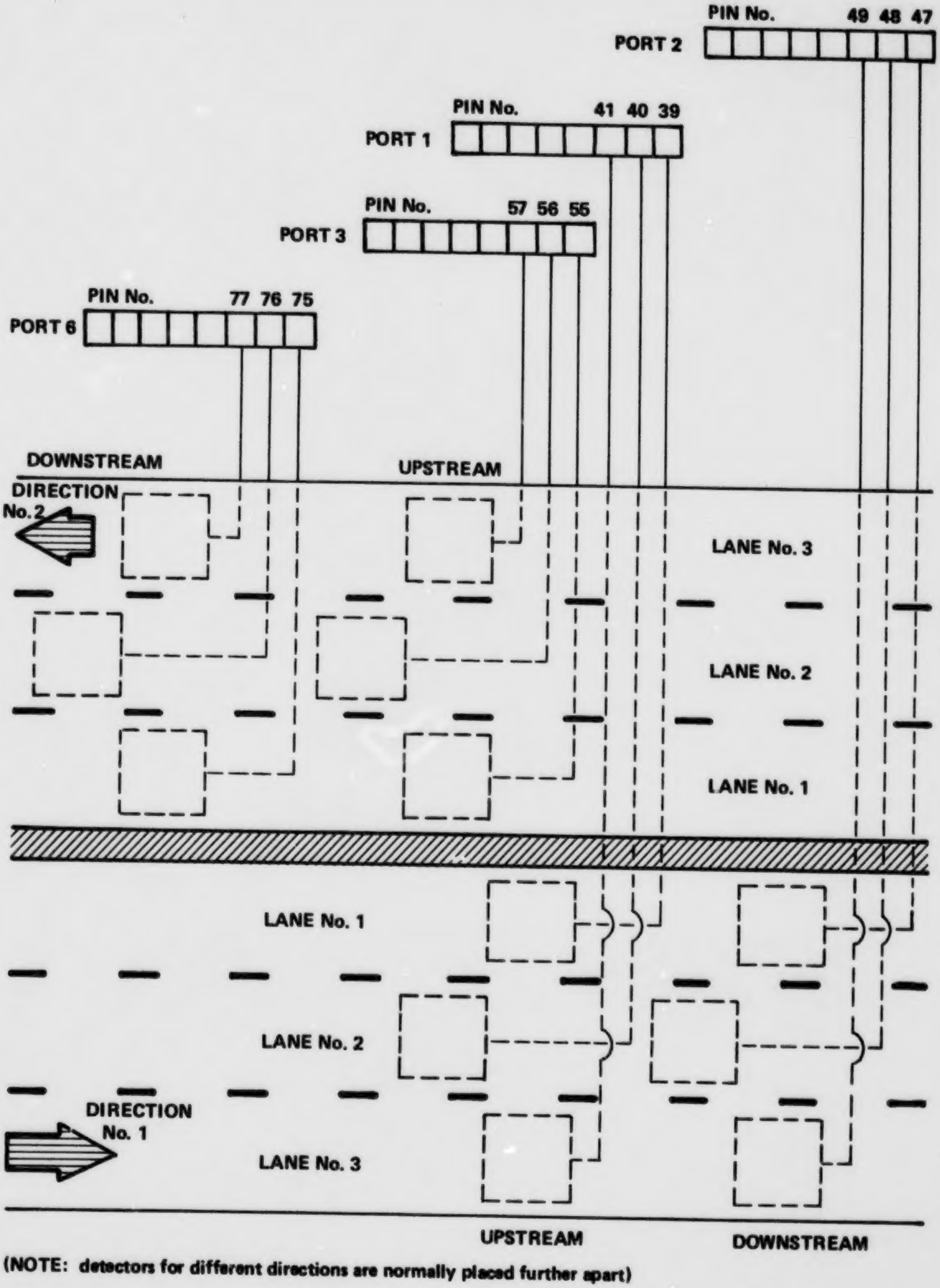
PIN No.	62	61	60	59	58	57	56	55
LANE No. (OR BIT No.)	8	7	6	5	4	3	2	1

DOWNSTREAM DETECTORS, DIRECTION No. 2

PORT 6 ($5006)

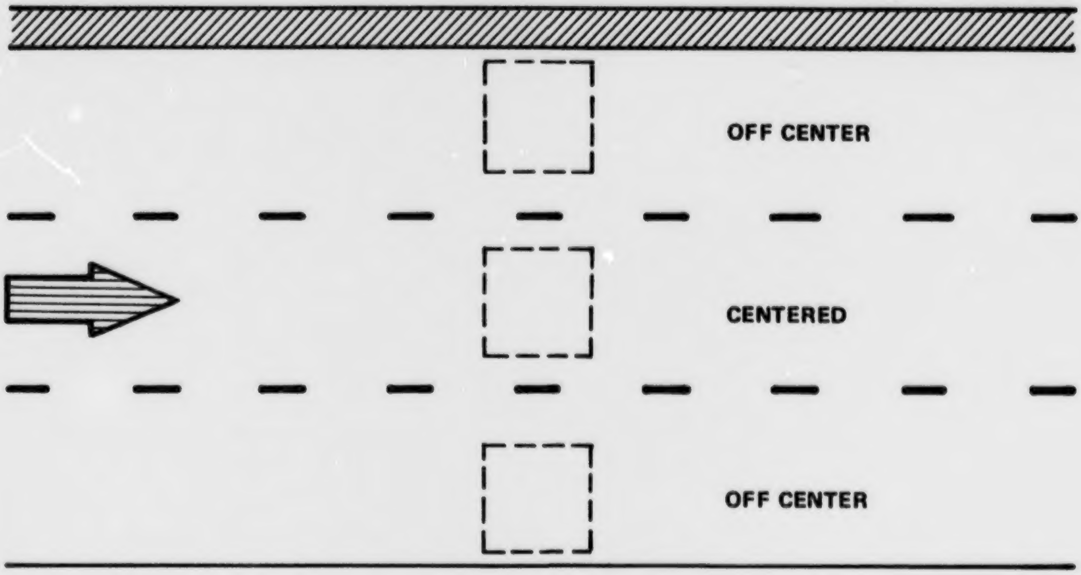
PIN No.	82	81	80	79	78	77	76	75
LANE No. (OR BIT No.)	8	7	6	5	4	3	2	1

Figure 7/ Detector Ports and Pin Assignments

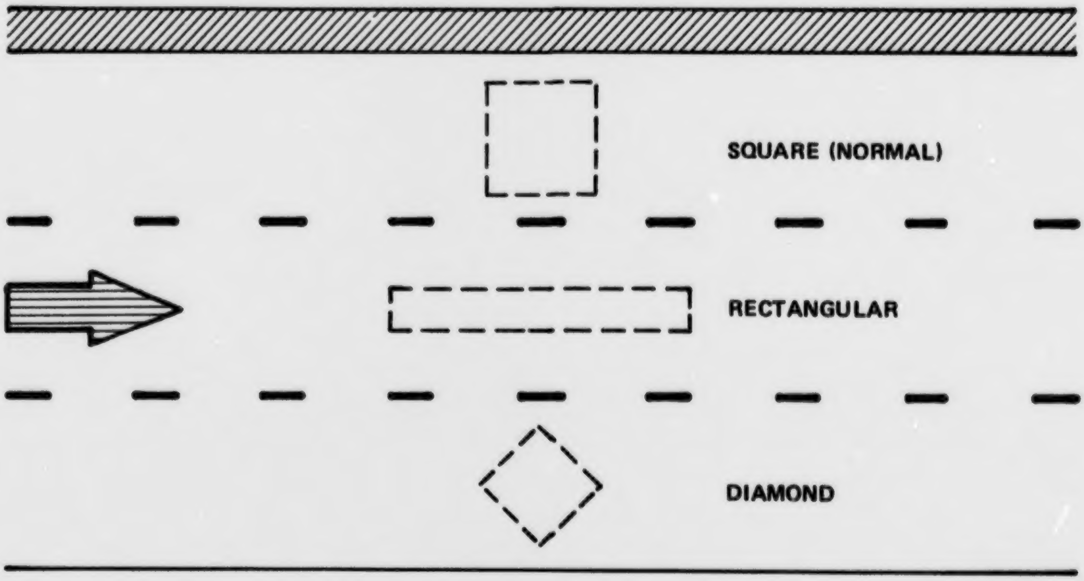

PIN No.
49 48 47
PORT 2
PIN No.
41 40 39
PORT 1
PIN No.
57 56 55
PORT 3
PIN No.
77 76 75
PORT 6
DOWNSTREAM
UPSTREAM
DIRECTION No. 2
LANE No. 3
LANE No. 2
LANE No. 1
LANE No. 1
LANE No. 2
DIRECTION No. 1
LANE No. 3
UPSTREAM
DOWNSTREAM

(NOTE: detectors for different directions are normally placed further apart)

Figure 8/ Sample Loop Configuration


OFF CENTER
CENTERED
OFF CENTER

(a) SPREAD LOOPS AS FAR APART AS POSSIBLE


SQUARE (NORMAL)
RECTANGULAR
DIAMOND

(b) USE DIFFERENT SHAPES

Figure 9/ Loop Installation Techniques

4/ DEFINITION OF DATA TO BE COLLECTED

The function of the PCS controller is to collect three types of information:

- volume counts,
- speed classification, and
- length counts.

This information, and the manner in which it is collected, is discussed in the following sections.

4.1/ Volume Counts

Using the upstream detectors, the field station simply counts the number of vehicles that travel along the lanes of the highway. Up to eight different volume counts can be specified at initialization time, with any lane (or lanes) being placed in any volume count. The only restriction is that lanes from two different directions cannot be included in the same volume count.

When a vehicle passes over a detector in a lane, the volume count associated with that lane is increased. Volume counts are collected for certain time intervals for every lane and, currently, these data are collected into hourly counts as specified during initialization. The data are currently kept for the last five days, but it is planned to expand this to a considerably longer time period as permitted by the available memory.

For example, the following counts could be defined:

- Count 1 -- number of vehicles using Lane 1 in Direction 1
- Count 2 -- number of vehicles using Lanes 2, 3 & 5 in Direction 1
- Count 3 -- number of vehicles using Lanes 1, 2, 3, 4 & 5 in Direction 2

These volume count definitions can be changed at any time from the central station or via the keyboard.

4.2/ Speed Classification

The speed of a vehicle can be calculated only in lanes with two detectors. By measuring the time between the triggering of the upstream and downstream detectors, and by using the known distance between detectors, the speed of a vehicle can be calculated. Each such vehicle is placed in one of the following 12 speed classes:

over 130 km/h
120-129 km/h
110-119 km/h
100-109 km/h
90- 99 km/h
80- 89 km/h
70- 79 km/h
60- 69 km/h
50- 59 km/h
40- 49 km/h
30- 39 km/h
under 30 km/h

Speed data are kept on a daily basis for the last five days (though it is planned to increase this where memory permits).

4.3/ Length Counts

As with speed classification, length counts apply only to lanes equipped with two detectors. The length of a vehicle is calculated from its speed (already computed), the time during which the vehicle activated a single detector, and the known width of the detector. If the length is greater than 6.5 m, the vehicle will be included in the length count for the current hour of the day.

Length counts are kept hourly for the last five days (though it is planned to increase this in future). The length counts are kept for precisely the same lane combinations as the volume counts.

5/ KEYBOARD OPERATIONS

5.1/ Overview

The keyboard provides a means of examining and changing the controller's memory during a visit to the field station. Although a great many functions can be performed through the keyboard (controller initialization, lane description, error detection, et cetera) it should be kept in mind that the only operations being performed are the reading and changing of memory.

5.2/ Display Modes

There are 5 basic display modes, as described below. Note that there are other displays, but they are merely transitions to one of the 5 modes.

5.2.1/ Blank Mode

Here, the entire display is blank, and all the display lights are turned off (see Figure 10). The controller reverts to a blank display several hours after a visit to a field station (more precisely, the display is blanked out 255 minutes after the last key was pressed). Also, the display is blanked out about 8 seconds after the controller is turned on (unless a key is pressed). Pressing any key while in blank mode causes the display to enter time mode.

5.2.2/ Time Mode

Here, the current time (on a 24-hour clock) is displayed -- hours on the upper display, minutes and seconds on the lower display. The day of the week is also displayed on the lights (Figure 11). This mode is entered when the keyboard user first presses a key either after power-up or after the display has been blanked out. Also, during the course of using the keyboard, the display reverts to time mode for the following reasons:

- The user has requested to change memory when he is not authorized to do so (see Changing Memory).

- The user has entered an unreadable address (see Examining Memory).
- The user has pressed a key whose specific function is to return to time mode.

5.2.3/ Hex Display Mode

When in time mode, the next four key pushes will be accepted and interpreted as a request to examine memory (see Figures 12 to 14), causing the display to enter hex display mode (note: leading zeros of an address must be entered). This mode displays (in hex) the data at the memory location entered by the user (see Figure 15).

From this mode the user can enter a new value, switch to hex/binary display, or revert to time mode.

5.2.4/ Hex/Binary Display Mode

This mode, like the hex display mode, displays the contents of memory (in hex) on the upper display. In addition, it displays the data in binary format on the display lights (see Figure 18). From this mode, the user can enter a new value, switch to hex display mode, revert to time mode, or toggle bits of memory (see Changing Memory). The hex/binary display mode is distinguished from the hex display mode by light 0. The former display turns light 0 on, while the latter display turns light 0 off.

5.2.5/ Power-Up Display

This mode displays, for 8 seconds, the results of the memory tests performed by the controller. (Refer to Figures 19 to 21.) This display is entered only when the controller is turned on.

If there are no memory errors, "good" is displayed (Figure 19) for 8 seconds, and the display is then blanked out.

If there are PROM errors but no RAM errors, display 9(b) of Figure 20 is displayed for 8 seconds and the display is then blanked out.

If there are RAM errors but no PROM errors, display 9(c) of Figure 21 is displayed for 8 seconds and the display is then blanked out.

If there are both RAM errors and PROM errors, display 9(b) is shown for 4 seconds, then display 9(c) is shown for 4 seconds and then the display is blanked out.

5.2.6/ Other Displays

All other displays are merely transitions between the five modes described above. The controller is waiting for input from the user, either a complete address or a complete data byte. The number of digits the user must enter is indicated by flashing lights -- one light flashes if one digit is required, two lights flash if two digits are required, or three lights flash to indicate three more digits are required (refer to Figures 12, 13, 14, 16 and 17.

5.3/ Different Displays and the Meaning of the Keys

The keys of the keyboard have different functions depending on which display is currently being shown. Figures 10 to 21 describe all the displays and the meaning of each key.

5.4/ Examining Memory

The keyboard user can examine the following:

- any RAM location (address $0000-maximum RAM available),
- any PROM location (address $C000-$FFFF), and
- detector ports (address $5001-$5006).

The latter two are of little use to the average keyboard user. PROM examinations are probably of use only to a system programmer, and detector port examinations are mainly of use to ensure detectors are working correctly.

The following descriptions indicate the means by which a user may examine memory.

i/ To examine memory from blank mode -- Press any key to enter time mode and proceed as in ii, below.

ii/ To examine memory from time mode -- Enter the 4-digit address, including all leading zeros (displays 2, 3, 4 and 5 will be shown, as applicable). The display will now be in hex display mode, unless the address entered is not a RAM, PROM, or detector port location -- in which case the display reverts to time mode.

iii/ To examine memory from hex display mode (i.e., to examine a different memory location) -- Push key 9 to revert to time mode and proceed as in ii, above. Actually, any of keys 0-9 can be used, but since key 9 also exits from hex/binary display, the user should make a habit of using the key 9. (Also, see Stepping Through Memory.)

iv/ To examine memory from hex/binary display mode -- Push key 9 to revert to time mode and proceed as in ii, above. (Also, see Stepping Through Memory.)

Once a memory location is displayed, the user can switch back and forth between hex display and hex/binary display by pressing the F key as needed.

5.5/ Changing Memory

To change a memory location, the user must be authorized to do so. Firstly, only changes to RAM are permitted (obviously). Secondly, to prevent unauthorized changes, a security password must be entered.

To set the password, the user must enter the value $07 at location $0001, and the value $05 at location $0002 (these two locations are not protected and can always be changed). To enter the password from time mode, use the following key sequence.

0001 -- (go to address 0001)
E07 -- (enter first part of password)
A -- (go to address 0002)
E05 -- (enter rest of password)
9 -- (return to time mode)

The security password is now set. It remains set for the entire keyboard session and is cleared about 4 hours after the last key is pressed.

Thirdly, the address cannot be a specially protected location, protected for programming reasons that need not concern the user.

If any of the above three conditions is not satisfied, an attempt to change memory is unauthorized and the display reverts to time mode.

5.5.1/ Entering New Hex Data

To enter a new hex value, do the following.

1/ Enter security password (unless it has recently been done).

2/ Display address to be changed (see Examining Memory) in either hex or hex/binary display.

3/ Press the E key (E for enter).

If the display reverts to time mode, the change is unauthorized (not RAM, password not set, or location is protected). Otherwise, two flashing lights will prompt the user for two digits (see Figure 16), and he can proceed.

4/ Enter two hex digits of data.

The two digits just entered will now appear on the upper display, unless the controller changed the memory location just after the new value was entered. After entering data using E, the display will always be in hex display mode. For example, to enter $8B at location $034C starting from time mode, use the following sequence:

```
0001E07AE059        -- (set password)
034C                -- (enter address)
E8B                 -- (enter new value)
```

5.5.2/ Changing Bits

The keyboard can be used to turn on or off specific bits at a memory location. As with entering a completely new value, no memory is changed unless the security password is set and the location to be changed is in RAM and is not specially protected. To change a bit, first display the data in hex/binary display mode (see Examining Memory). Light 0 is now on to indicate hex/binary display. Now, if the light corresponding to a bit is on (bit 1 for light 1, ..., bit 8 for light 8), then that bit is set. If the light is off, that bit is reset (see Figure 18). To toggle bit n (n = 1, 2, ..., 8), push key n. If the bit was reset it is now set and vice versa, unless the change is not authorized, in which case the display reverts to time mode.

For example, to reset bit 3 of location $002E starting from time mode, use the following key sequence:

```
        0001E07AE059        -- (enter password)
        002E                -- (display data)
        F                   -- (switch to hex/binary display)
```

If light 3 is off, the bit is already reset so enter:

```
        9                   -- (return to time mode)
```

If light 3 is on, the bit is now set:

```
        3                   -- (turn off the bit)
        9                   -- (return to time mode)
```

5.5.3/ Clearing Memory

To zero out a memory location, one can use "E00" or toggle all the bits so that the value is 0. A faster way, however, can be used in the hex/-binary display mode (in which light 0 is on). Of course, changes must be authorized, or the display reverts to time mode.

To clear a location, display it (in the usual way) and enter hex/binary display mode. Then press 0, and the memory is cleared (if the change is authorized). Used in conjunction with memory stepping functions (see Stepping Through Memory) this facilitates a fast method of clearing memory. For example, to clear memory locations $0257 to $025A starting from time mode, use the following key sequence:

0001E07AE059 -- (set security password)
0257F -- (display $0257 in hex/binary)
0 -- (clear it)
A -- (move to next location - $0258)
0 -- (clear it)
A -- (move to next location - $0259)
0 -- (clear it)
A -- (move to next location - $025A)
0 -- (clear it)
9 -- (return to time mode)

5.5.4/ Protected Locations

The following hexadecimal RAM locations are protected and cannot be changed:

- system stack -- 32 bytes at the end of RAM (e.g., for a 4K controller, the stack occupies addresses $0FE0-$0FFF)
- the following special locations -- 0020, 0033, 0034, 0035, 0038, 0039, 003A, 003B, 006E, 006F

Note that page 0, column 2 locations $0020-$002D are used by the keyboard and display handling routines. If these locations are changed, strange things can happen to the display. This can be corrected by pressing the 9 key until the display reverts to time mode.

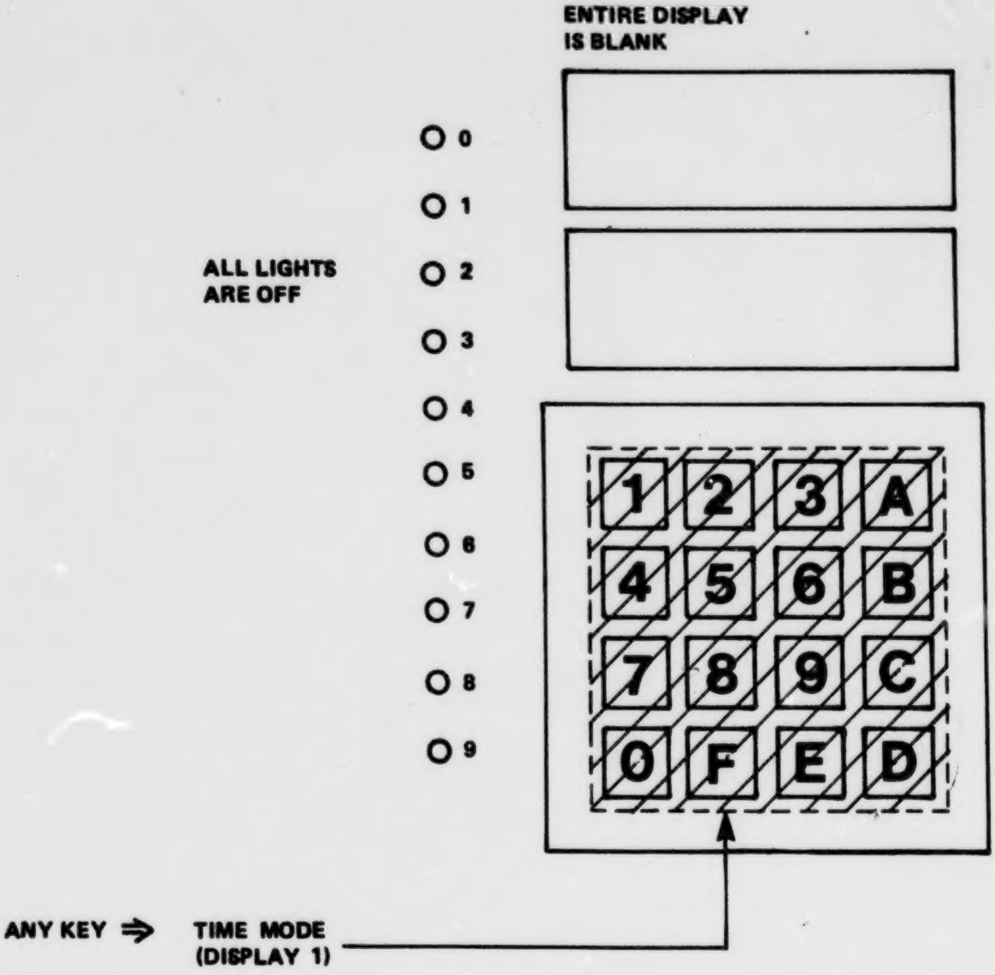
5.6/ Stepping Through Memory

In either hex display mode or hex/binary display mode, the keys A, B, C and D can be used to quickly step through memory. The keys have the following functions.

A -- move adress ahead by 1
B -- move address back by 1

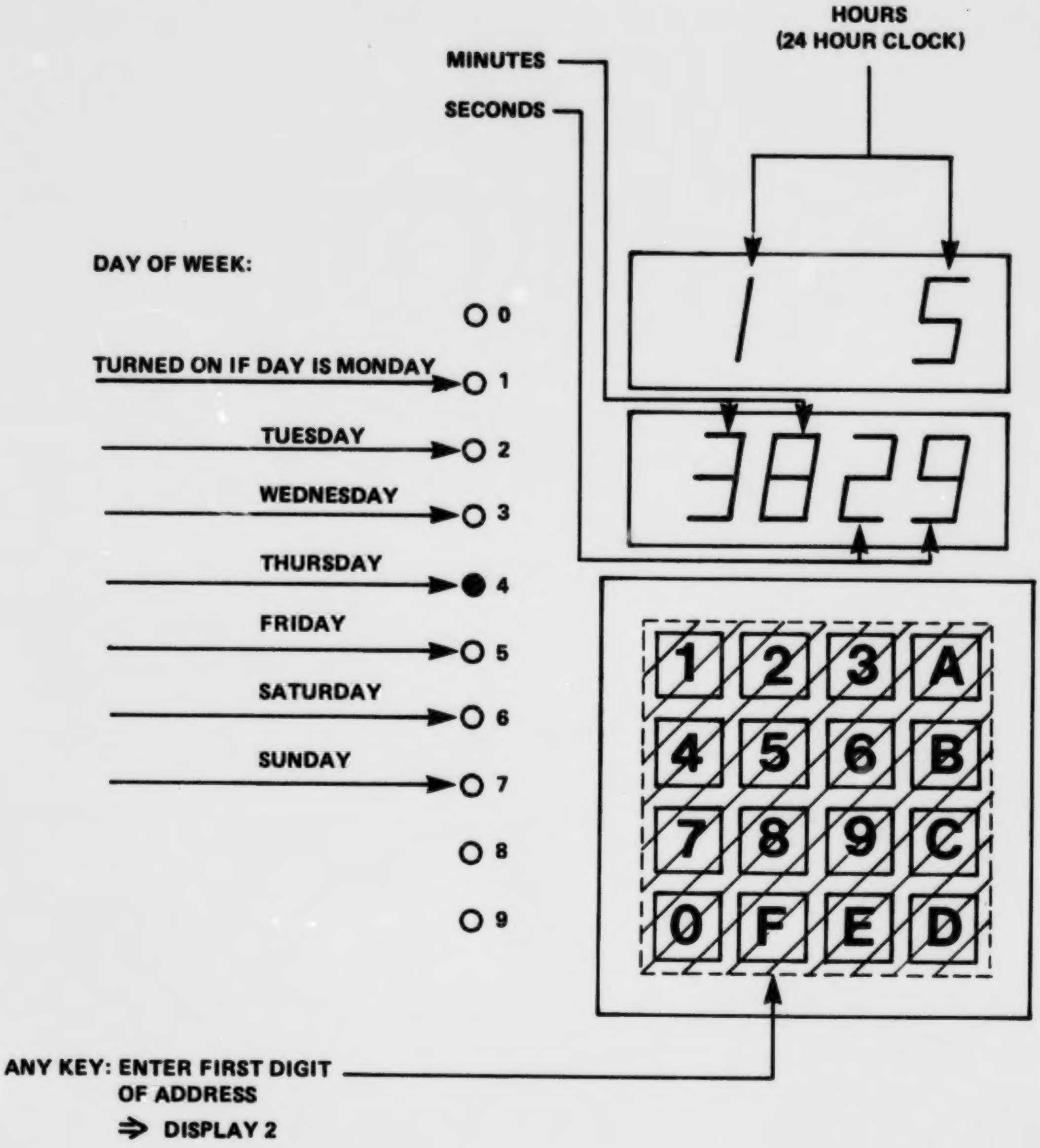
C -- move address ahead by $10
D -- move address back by $10

If the new address is not a RAM, PROM, or detector port location, the display reverts to time mode. Otherwise, the new address and data are displayed.


ENTIRE DISPLAY IS BLANK
ALL LIGHTS ARE OFF
0
1
2
3
4
5
6
7
8
9
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
ANY KEY ⇒ TIME MODE (DISPLAY 1)

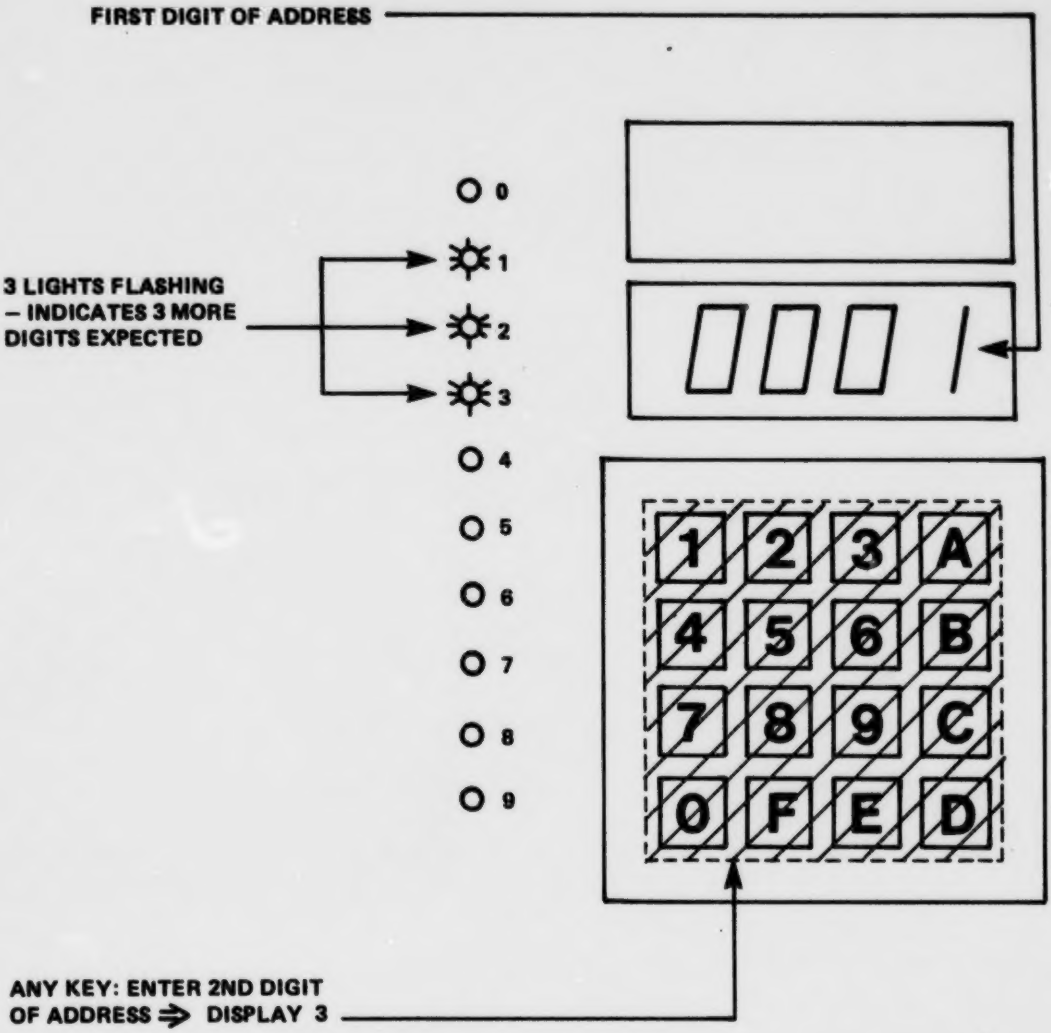
NOTE: ANY DISPLAY WILL REVERT TO THE BLANK DISPLAY IF NO KEY IS PRESSED FOR MORE THAN 255 MINUTES.

Figure 10/ Display 0 — Blank Display


HOURS
(24 HOUR CLOCK)
MINUTES
SECONDS
DAY OF WEEK:
0
TURNED ON IF DAY IS MONDAY
1
TUESDAY
2
WEDNESDAY
3
THURSDAY
4
FRIDAY
5
SATURDAY
6
SUNDAY
7
8
9
1 5
3829
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
ANY KEY: ENTER FIRST DIGIT
OF ADDRESS
⇒ DISPLAY 2

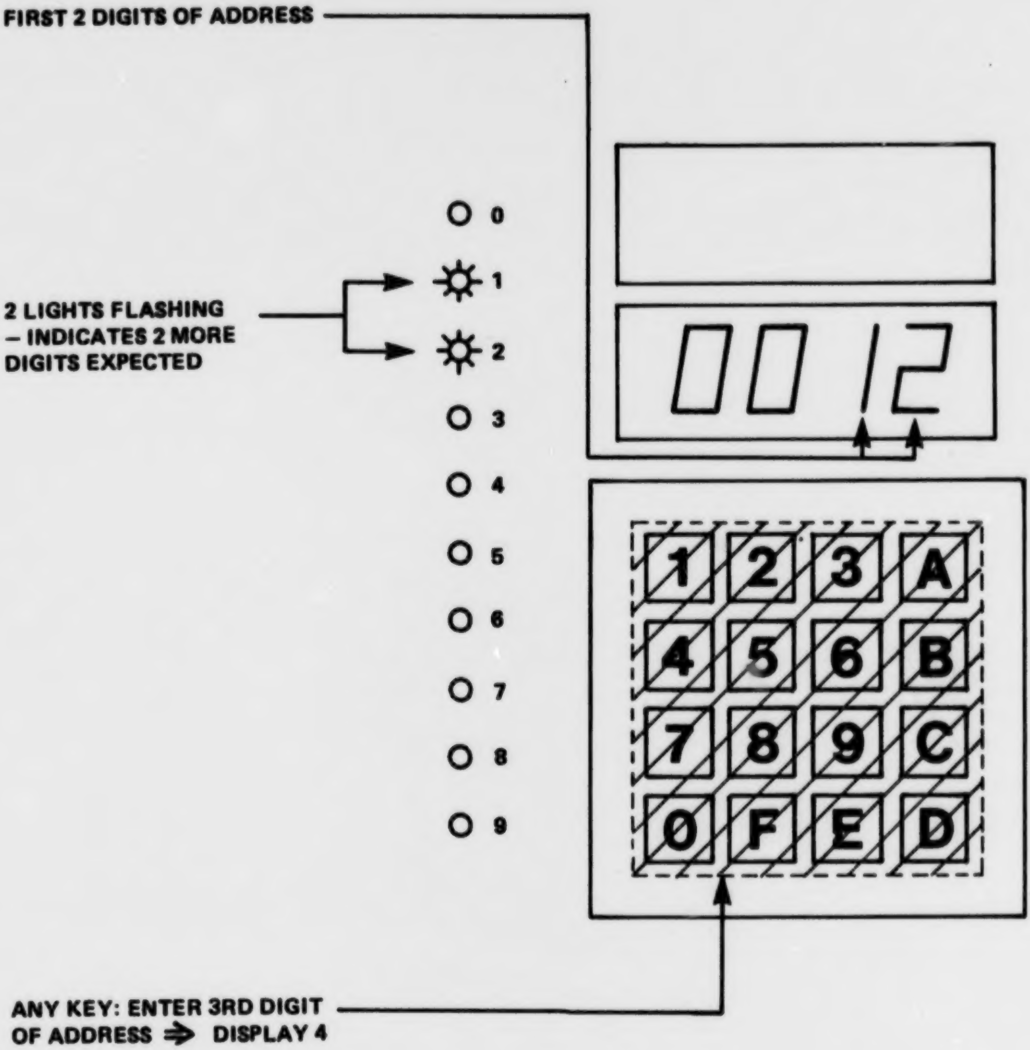
(SAMPLE DISPLAY FOR 3:38:29 p.m., THURSDAY)

Figure 11/ Display 1 – Time Mode


FIRST DIGIT OF ADDRESS
3 LIGHTS FLASHING
– INDICATES 3 MORE
DIGITS EXPECTED
0
1
2
3
4
5
6
7
8
9
0001
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
ANY KEY: ENTER 2ND DIGIT
OF ADDRESS ⇒ DISPLAY 3

(SAMPLE FOR DISPLAY OF LOCATION $1234)

Figure 12/Display 2 — First Digit of Address Has Been Entered



FIRST 2 DIGITS OF ADDRESS

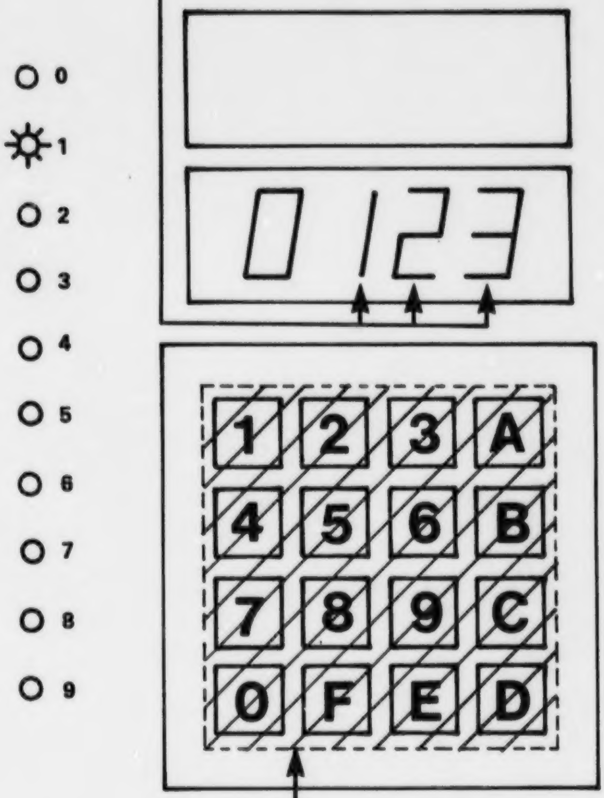
2 LIGHTS FLASHING
– INDICATES 2 MORE
DIGITS EXPECTED

ANY KEY: ENTER 3RD DIGIT
OF ADDRESS ⇛ DISPLAY 4

0
1
2
3
4
5
6
7
8
9
00 12
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D

(SAMPLE FOR DISPLAY OF LOCATION $1234)

Figure 13/ Display 3 – Two Digits of Address Have Been Entered


0
1
2
3
4
5
6
7
8
9
0123
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D

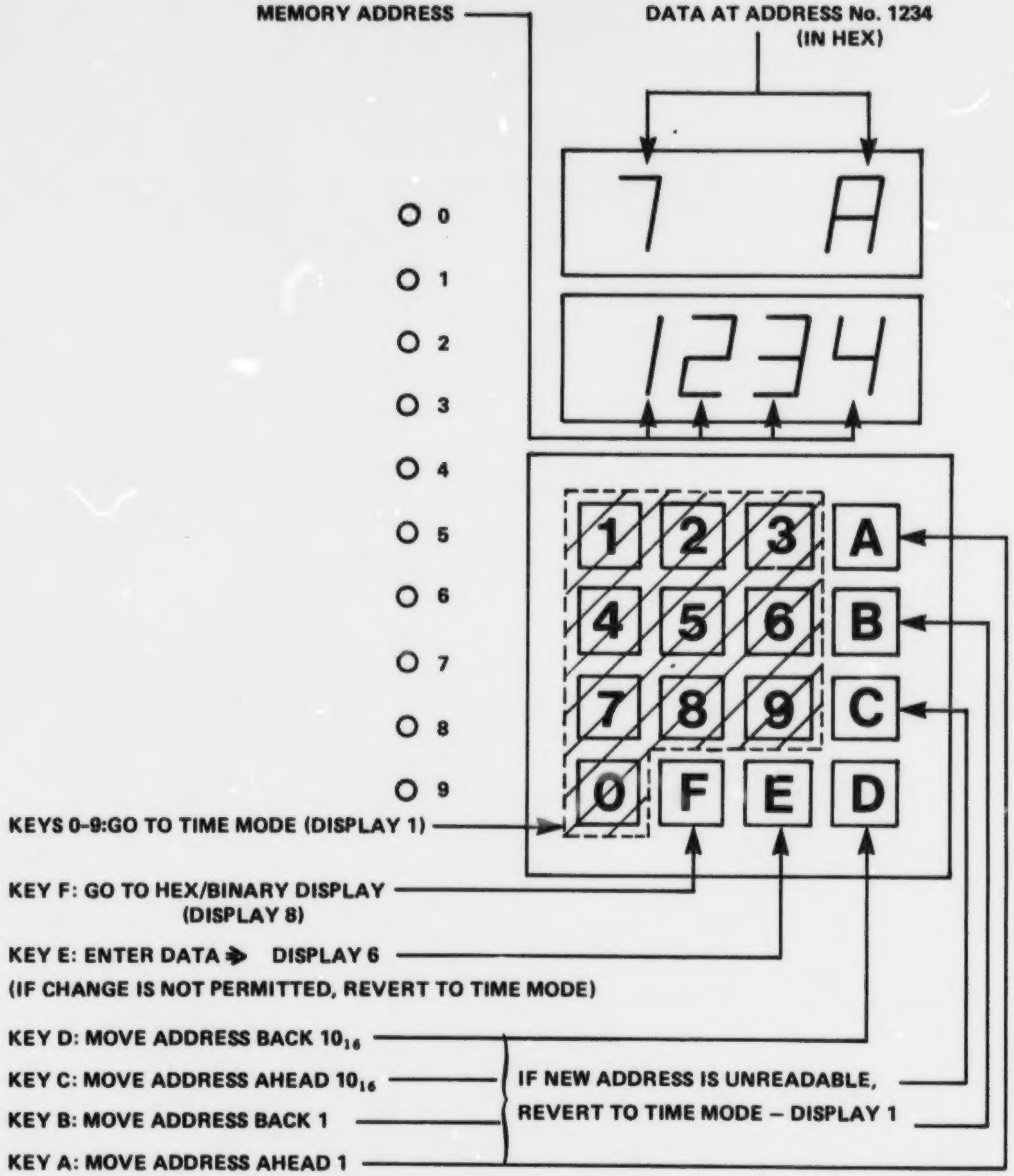
FIRST 3 DIGITS OF ADDRESS

FLASHING TO INDICATE
1 MORE DIGIT IS EXPECTED

ANY KEY: ENTER 4TH DIGIT
OF ADDRESS ⇛ DISPLAY 5
(UNLESS AN UNREADABLE ADDRESS IS ENTERED, IN WHICH CASE THE DISPLAY
REVERTS TO TIME MODE – DISPLAY 1)

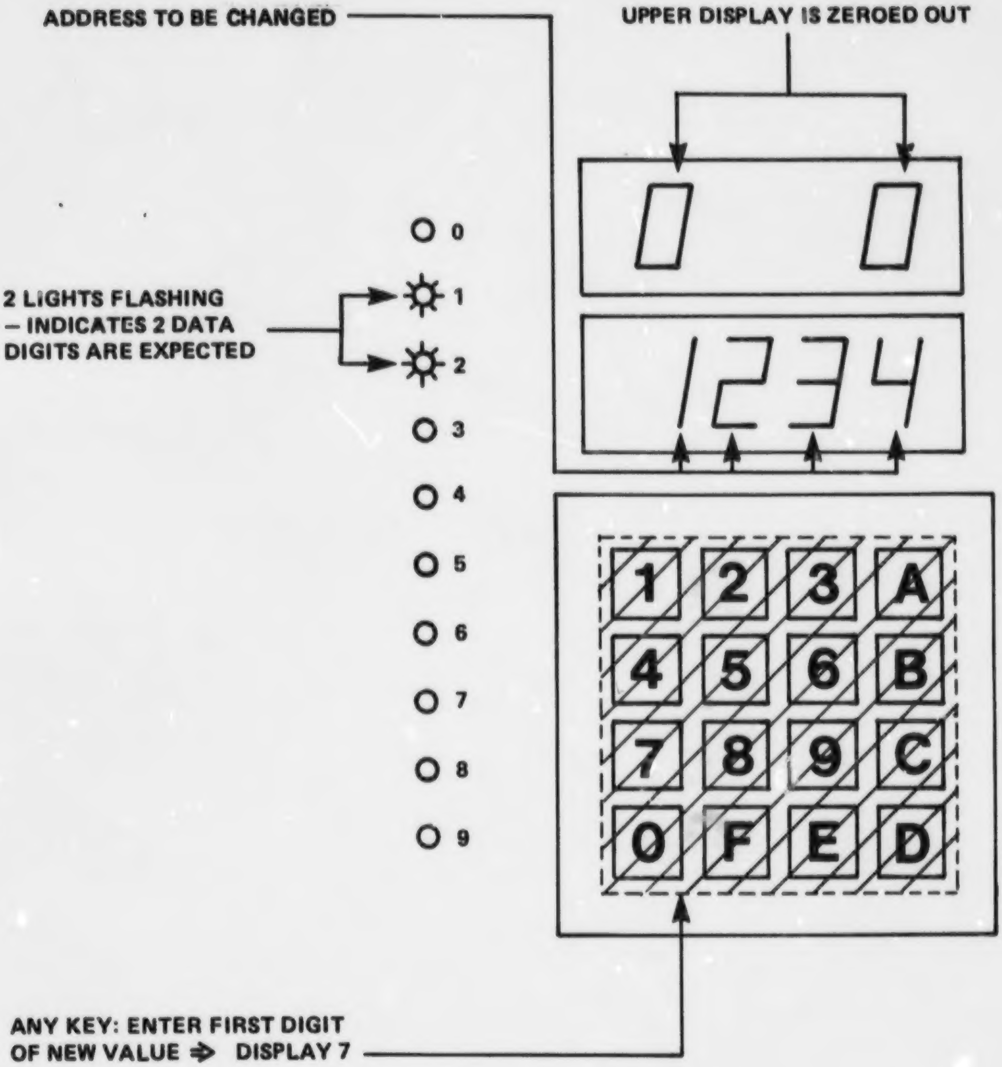
(SAMPLE FOR DISPLAY OF LOCATION $1234)

Figure 14/ Display 4 – Three Digits of Address Have Been Entered


MEMORY ADDRESS
DATA AT ADDRESS No. 1234 (IN HEX)
7 A
1234
0
1
2
3
4
5
6
7
8
9
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
KEYS 0-9:GO TO TIME MODE (DISPLAY 1)
KEY F: GO TO HEX/BINARY DISPLAY (DISPLAY 8)
KEY E: ENTER DATA ➔ DISPLAY 6
(IF CHANGE IS NOT PERMITTED, REVERT TO TIME MODE)
KEY D: MOVE ADDRESS BACK 10_{16}
KEY C: MOVE ADDRESS AHEAD 10_{16}
KEY B: MOVE ADDRESS BACK 1
KEY A: MOVE ADDRESS AHEAD 1
IF NEW ADDRESS IS UNREADABLE,
REVERT TO TIME MODE – DISPLAY 1

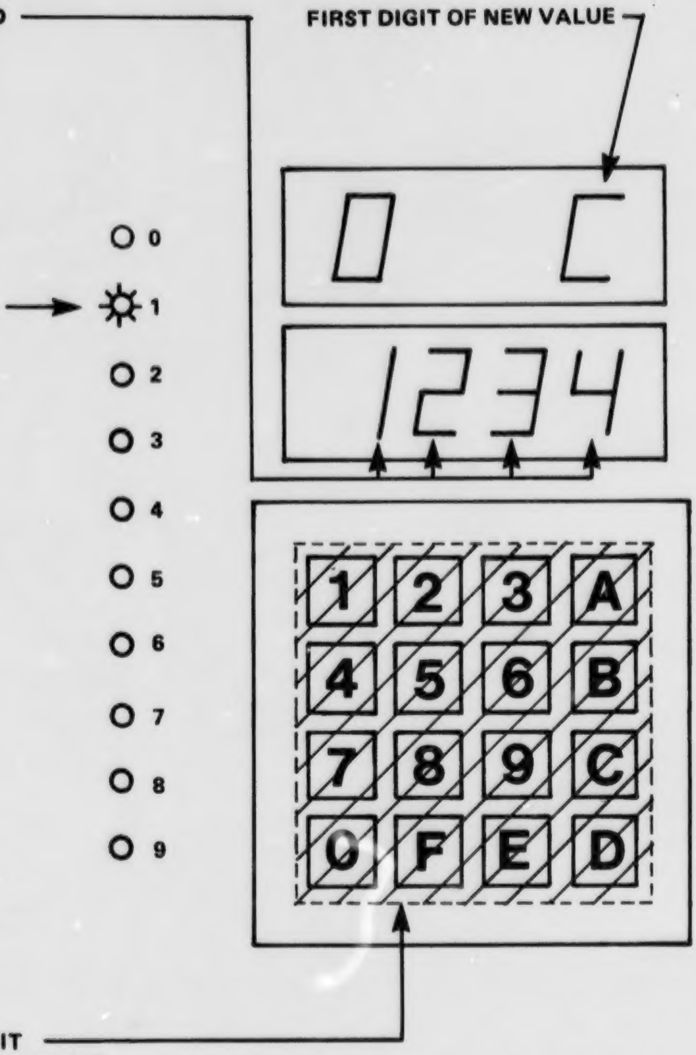
(SAMPLE FOR DISPLAY OF LOCATION $1234)

Figure 15/Display 5 – Hex Display


ADDRESS TO BE CHANGED
UPPER DISPLAY IS ZEROED OUT
0 0
1234
0
1
2
3
4
5
6
7
8
9
2 LIGHTS FLASHING – INDICATES 2 DATA DIGITS ARE EXPECTED
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
ANY KEY: ENTER FIRST DIGIT OF NEW VALUE ⇛ DISPLAY 7

(SAMPLE DISPLAY FOR STORING $CA TO LOCATION $1234)

Figure 16/ Display 6 – Enter Hex Value (1st Digit)


0
C
1234
0
1
2
3
4
5
6
7
8
9
1
2
3
A
4
5
6
B
7
8
9
C
0
F
E
D

ADDRESS TO BE CHANGED

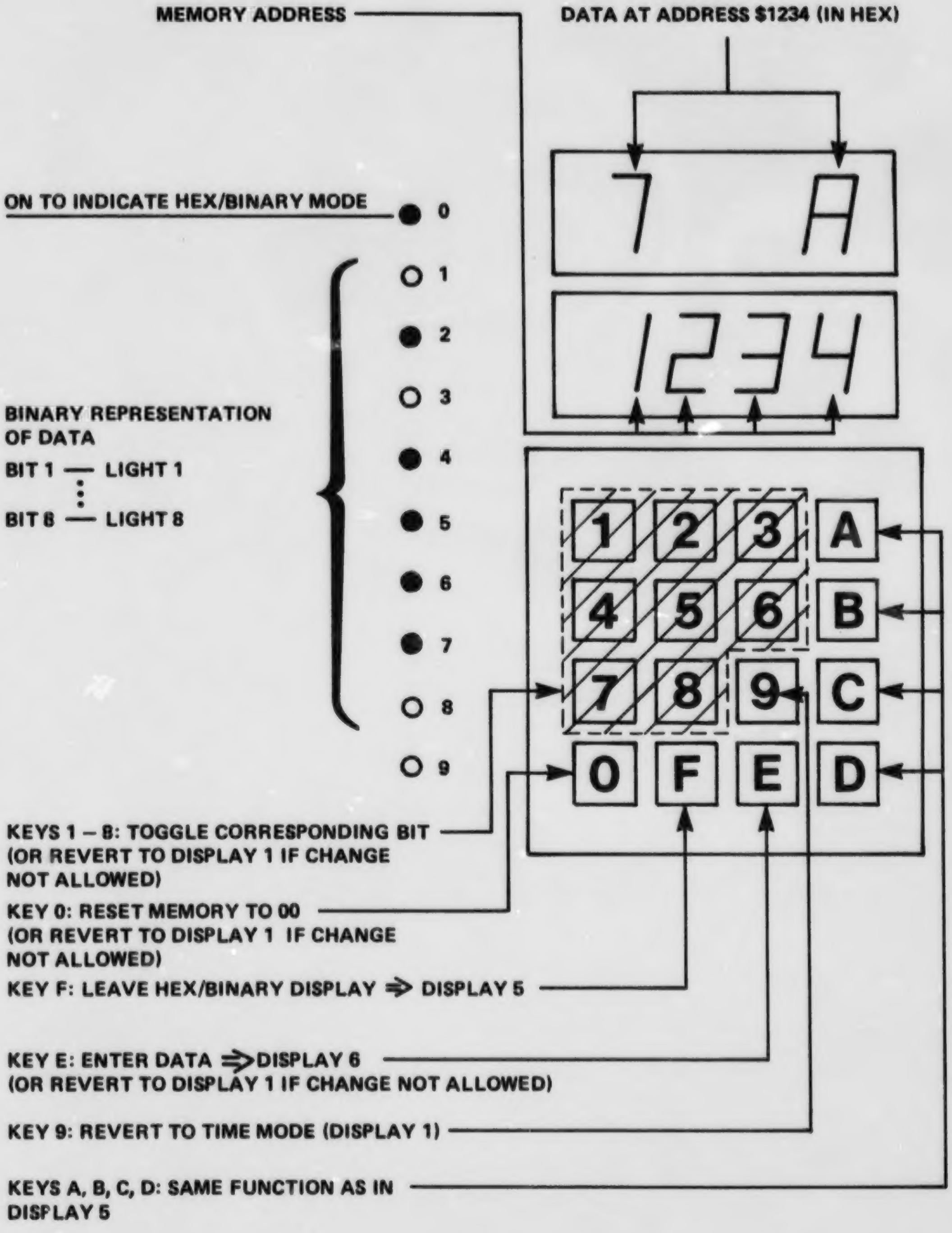
FIRST DIGIT OF NEW VALUE

1 LIGHT FLASHING
– INDICATES 1 MORE DIGIT
OF DATA IS EXPECTED

ANY KEY: ENTER LAST DIGIT
OF NEW DATA VALUE ⇒ DISPLAY 5

(SAMPLE DISPLAY FOR STORING $CA TO LOCATION $1234)

Figure 17/ Display 7 – Enter Hex Value (2nd Digit)


MEMORY ADDRESS
DATA AT ADDRESS $1234 (IN HEX)
7 A
1234
ON TO INDICATE HEX/BINARY MODE
0
1
2
3
4
5
6
7
8
9
BINARY REPRESENTATION OF DATA
BIT 1 — LIGHT 1
BIT 8 — LIGHT 8
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D

KEYS 1 – 8: TOGGLE CORRESPONDING BIT (OR REVERT TO DISPLAY 1 IF CHANGE NOT ALLOWED)

KEY 0: RESET MEMORY TO 00 (OR REVERT TO DISPLAY 1 IF CHANGE NOT ALLOWED)

KEY F: LEAVE HEX/BINARY DISPLAY ⇒ DISPLAY 5

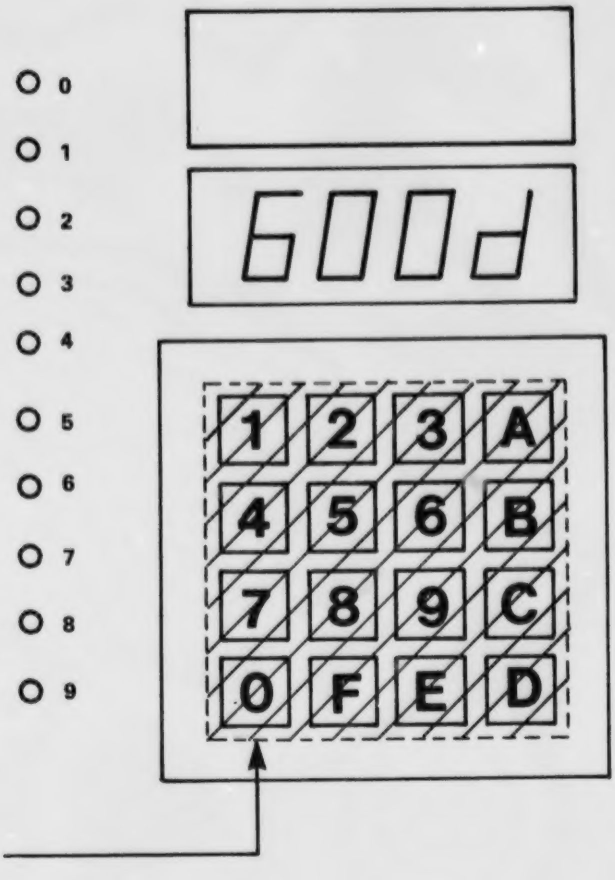
KEY E: ENTER DATA ⇒ DISPLAY 6 (OR REVERT TO DISPLAY 1 IF CHANGE NOT ALLOWED)

KEY 9: REVERT TO TIME MODE (DISPLAY 1)

KEYS A, B, C, D: SAME FUNCTION AS IN DISPLAY 5

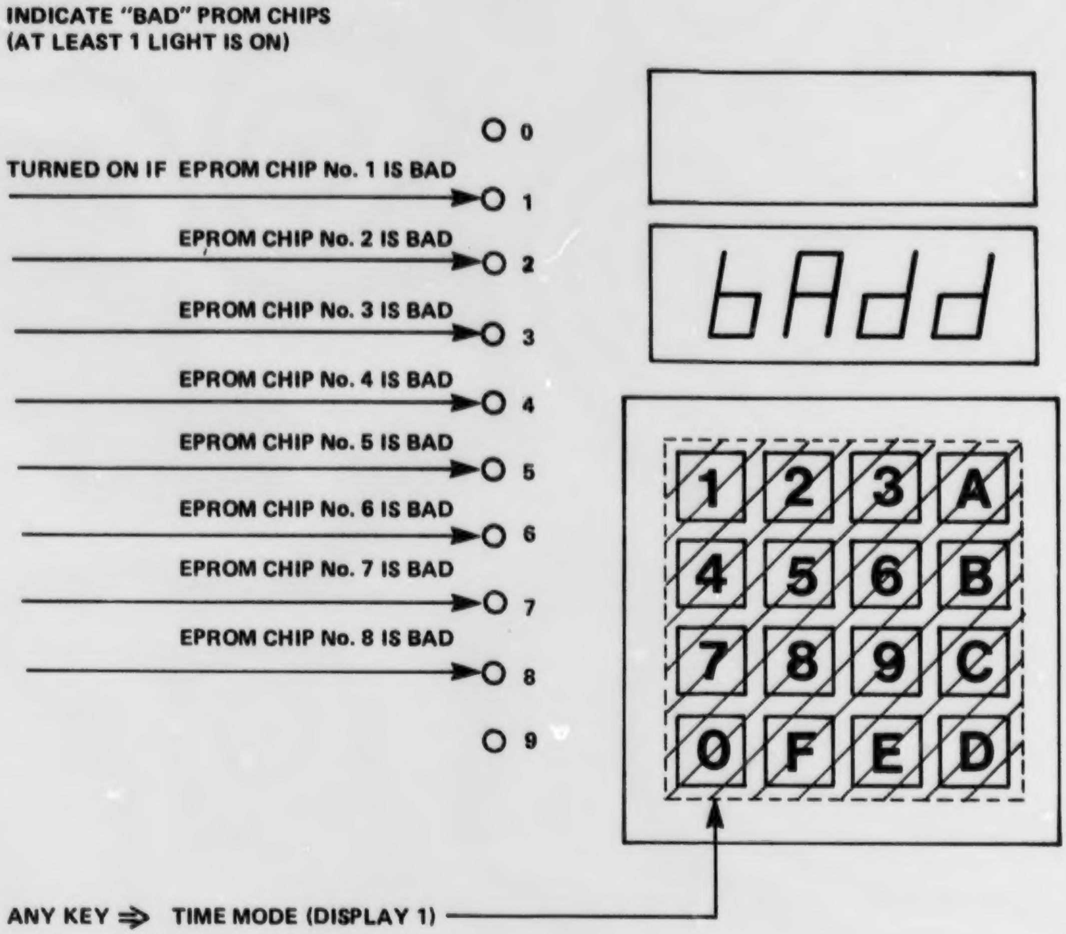
(SAMPLE FOR DISPLAY OF LOCATION $1234)

Figure 18/ Display 8 — Hex/Binary Display


0
1
2
3
4
5
6
7
8
9
600d
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
ANY KEY ⇒ TIME MODE (DISPLAY 1)

THIS DISPLAY IS SHOWN AFTER TURNING ON THE CONTROLLER IF THERE ARE NO MEMORY ERRORS. THE DISPLAY REMAINS FOR 8 SECONDS BEFORE BEING BLANKED OUT (DISPLAY0) UNLESS A KEY IS PUSHED.

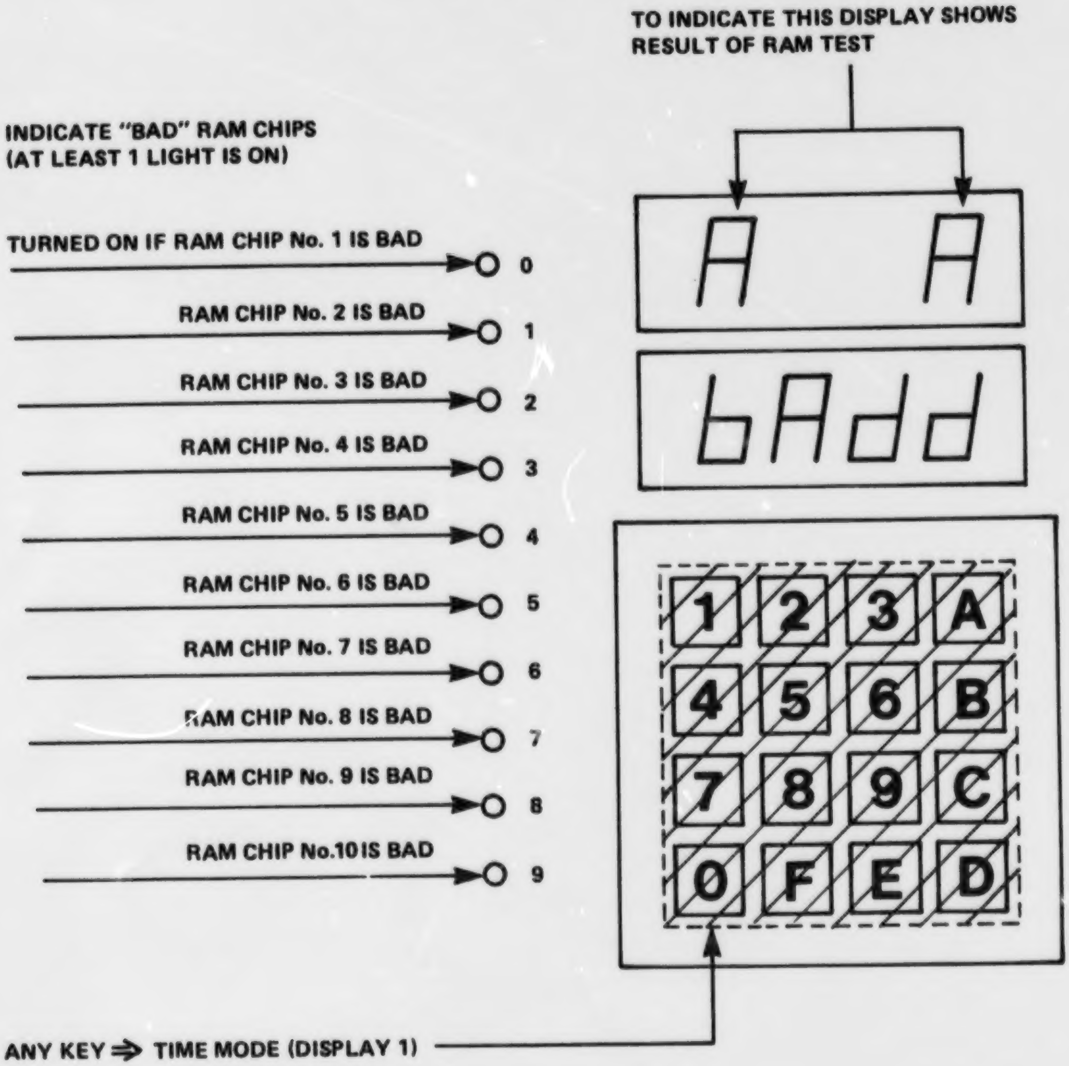
Figure 19/ Display 9(a) – Show Result of Memory Test


INDICATE "BAD" PROM CHIPS
(AT LEAST 1 LIGHT IS ON)
0
TURNED ON IF EPROM CHIP No. 1 IS BAD
1
EPROM CHIP No. 2 IS BAD
2
EPROM CHIP No. 3 IS BAD
3
EPROM CHIP No. 4 IS BAD
4
EPROM CHIP No. 5 IS BAD
5
EPROM CHIP No. 6 IS BAD
6
EPROM CHIP No. 7 IS BAD
7
EPROM CHIP No. 8 IS BAD
8
9
bAdd
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
ANY KEY ⇛ TIME MODE (DISPLAY 1)

THIS DISPLAY IS SHOWN AFTER TURNING ON THE CONTROLLER IF THERE ARE ANY PROM ERRORS. THE DISPLAY WILL APPEAR FOR 8 SECONDS (OR 4 SEC. IF THERE IS A RAM ERROR) BEFORE BEING BLANKED OUT (OR SHOWING ANY RAM ERRORS)

NOTE: CHIP No. 1 ADDRESSES $C000 – $C7FF
ETC:
CHIP No. 8 ADDRESSES $F800 – $FFFF

Figure 20/ Display 9(b) – Show PROM Errors


TO INDICATE THIS DISPLAY SHOWS RESULT OF RAM TEST
INDICATE "BAD" RAM CHIPS (AT LEAST 1 LIGHT IS ON)
A A
bAdd
TURNED ON IF RAM CHIP No. 1 IS BAD
0
RAM CHIP No. 2 IS BAD
1
RAM CHIP No. 3 IS BAD
2
RAM CHIP No. 4 IS BAD
3
RAM CHIP No. 5 IS BAD
4
RAM CHIP No. 6 IS BAD
5
RAM CHIP No. 7 IS BAD
6
RAM CHIP No. 8 IS BAD
7
RAM CHIP No. 9 IS BAD
8
RAM CHIP No.10 IS BAD
9
1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
ANY KEY ⇒ TIME MODE (DISPLAY 1)

THIS DISPLAY IS SHOWN AFTER THE CONTROLLER IS TURNED ON (AND AFTER ANY PROM ERRORS ARE DISPLAYED) IF THERE ARE ANY RAM ERRORS. THE DISPLAY REMAINS FOR 8 SECONDS, OR UNTIL A KEY IS PRESSED (OR 4 SECONDS IF THERE WERE PROM ERRORS).

NOTE: CHIP No. 1 ADDRESSES $0000 – $07FF
ETC.
CHIP No. 10 ADRESSES $4800 – $4FFF

Figure 21/ Display 9(c) – Show RAM Errors

6/ CONTROLLER INITIALIZATION

The following sections describe the means by which the controller is initialized from the keyboard. The same initialization can be performed from the central station using a special program which performs the same steps as required for a manual initialization. This program uses data from a disk file associated with a particular station.

To make the descriptions clear, they are accompanied by examples for a particular station, say Station 21. This is a theoretical station designed to illustrate the process of initialization. It is not necessarily realistic.

The lane detector configuration for the hypothetical station is shown in Figure 22. Note that one lane has no detector and another lane has only one detector. There are 6 lanes in each direction.

To initialize a station, the steps outlined in Sections 6.1 to 6.7 should be performed in the given order.

6.1/ Set Security Password

From time mode, enter the following key sequence (if the display is blank, first press any key to show the time).

```
0001E07  -- (enter $07 in location $0001)
AE05     -- (enter $05 in location $0002)
9        -- (revert to time mode)
```

The security password is now set and the display is in time mode.

6.2/ Set Station Address

Convert the station number (0-31) to its hexadecimal equivalent ($0-$1F). Enter this value in location 0000. For example, our sample station is Station 21, which is $15 in hexadecimal, so we enter:

```
0000E15  --  (enter station address)
9        --  (revert to time mode)
```

6.3/ Set Date and Time

The date and time variable locations ($0003-$000C) are shown in Figure 23 and should be initialized to the current date and time. These values are all binary coded decimal values.

Suppose the sample station is being initialized on Tuesday, November 9, 1982 (the 313th day of the year) and it is 10:08 in the morning. The following key sequence (from time mode) would enter the date and time.

```
0003E82 -- (enter year - 82)
AE11    -- (enter month -11)
AE09    -- (enter day - 9)
AE03    -- (enter first part of 0313)
AE13    -- (enter 2nd part of 0313)
AE02    -- (enter day of week, 2 = Tuesday)
AE10    -- (enter hour - 10)
AE08    -- (enter minutes - 8)
AE00    -- (enter seconds - 0)
9       -- (return to time mode)
```

There is no great need to set the 1/10 second counter. The other locations in the figure ($000D-$000F) need not be initialized at all.

6.4/ Define Station Geometry

The locations $0010-$0015 (see Figure 24) are used to define the roadway and loop detector arrangement, and they must be initialized. Since these descriptive bytes map bits onto lanes, the hex/binary display mode facilitates lane description.

The following key sequence (from time mode) would be used for the sample station (refer to Figure 22).

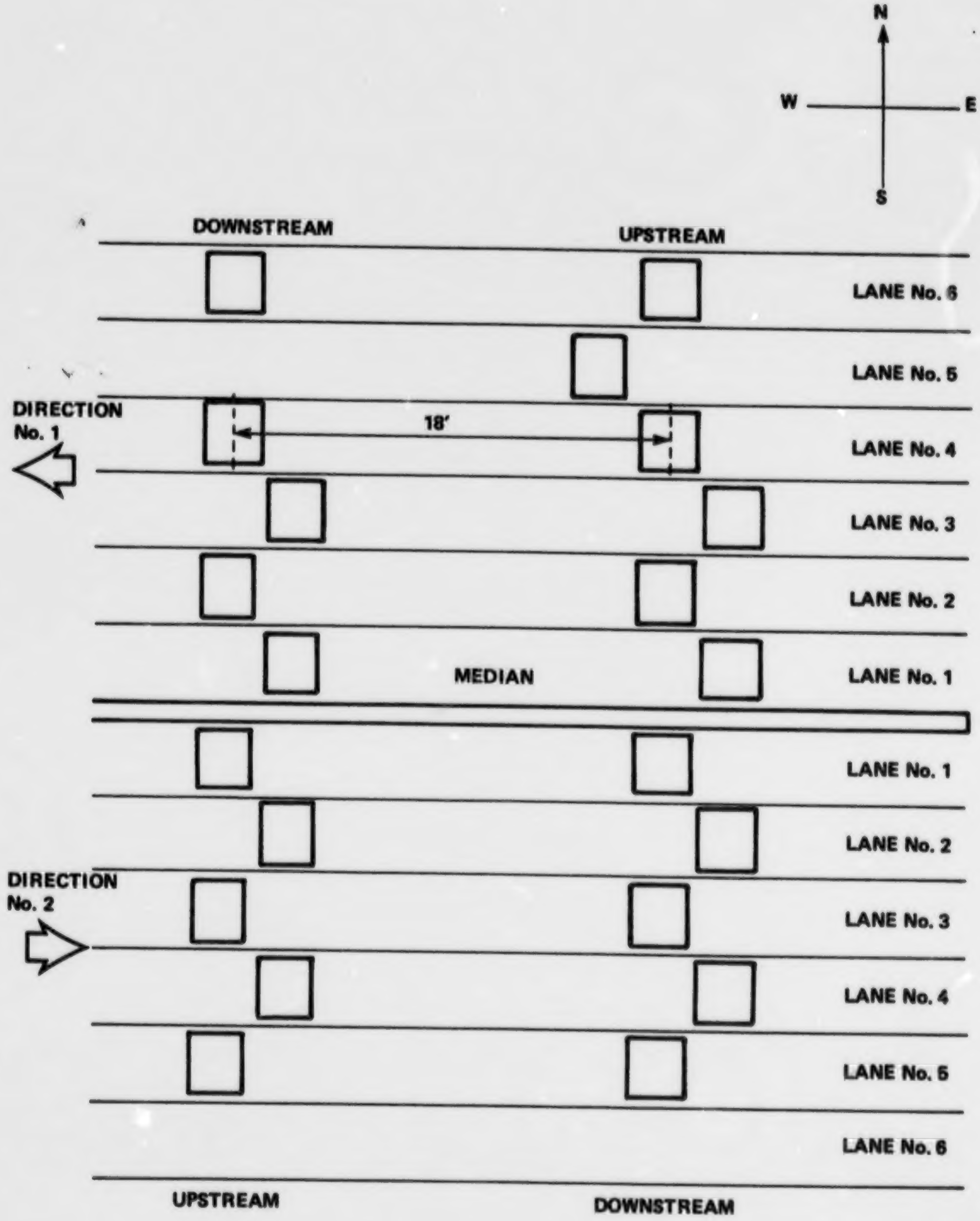

N
W
E
S
DOWNSTREAM
UPSTREAM
LANE No. 6
LANE No. 5
DIRECTION
No. 1
18'
LANE No. 4
LANE No. 3
LANE No. 2
MEDIAN
LANE No. 1
LANE No. 1
LANE No. 2
DIRECTION
No. 2
LANE No. 3
LANE No. 4
LANE No. 5
LANE No. 6
UPSTREAM
DOWNSTREAM

Figure 22/ Sample Roadway with Loop Detectors

```
PAGE  002  EQUATE  .SA:1

00024                   *
00025                   *
00026                   *        PAGE 0 COLUMN 0    MAIN VARIABLES
00027                   *
00028                   *
00029          0000  A PCSCAD EQU    $00      PCS CONTROLLER ADDRESS VALUE (0 TO 31) - 5 BITS , RIGHT JUSTIFIED
00030          0001  A SECCOD EQU    $01      SECURITY CODE  (= 0705 - 2 BYTES) PROTECTS RAM FROM UNAUTHORIZED CHANGES
00031          0003  A DATE   EQU    $03      CURRENT DATE, 3 BYTES - (YEAR, MONTH, DAY)
00032          0003  A YEAR   EQU    $03      CURRENT CALENDAR YEAR OF CENTURY  (BCD)
00033          0004  A MONTH  EQU    $04      CURRENT CALENDAR MONTH  (BCD)
00034          0005  A DAOFMO EQU    $05      CURRENT DAY OF MONTH  (BCD)
00035          0006  A DOYM   EQU    $06      CURRENT DAY OF YEAR - MSB  (BCD)
00036          0007  A DOYL   EQU    $07      CURRENT DAY OF YEAR - LSB  (BCD)
00037                   *
00038          0008  A DAOFWK EQU    $08      DAY OF WEEK (1= MONDAY)
00039          0009  A HR     EQU    $09      HOUR OF DAY - 24 HR CLOCK  (BCD: 0-23)
00040          000A  A MIN    EQU    $0A      CURRENT MINUTE  (BCD: 0-59)
00041          000B  A SEC    EQU    $0B      CURRENT SECOND  (BCD: 0-59)
00042          000C  A TSEC   EQU    $0C      1/10 SECOND COUNTER (0-9)
00043                   *
00044                   *        THE FOLLOWING VARIABLES ARE UPDATED PERIODICALLY -
00045                   *        NO MANUAL INITIALIZATION IS NECESSARY
00046                   *
00047          000D  A DIM    EQU    $0D      NUMBER OF DAYS IN CURRENT MONTH (BCD)
00048          000E  A DAYPTR EQU    $0E      BIT POINTER FOR DAY-OF-WEEK  (LSB=MON.)
00049          000F  A TIMEFX EQU    $0F      TIME CORRECTION FLAG-TO PERFORM DAYLIGHT
00050                   *                     SAVING CHANGE (LAST SUNDAY APRIL/OCTOBER 2AM)
00051                   *                     =0  CHANGE ALREADY DONE
00052                   *                     >0  CHANGE TO BE DONE
00053                   *
```

Figure 23/ Date and Time Locations

```
PAGE  003  EQUATE  .SA:1

00055                      *
00056                      *
00057                      *
00058                      *          PAGE 0 COLUMN 1    STATION GEOMETRY & VOLUME DESCRIPTION VARIABLES
00059                      *
00060                      *
00061          0010  A NLANE1 EQU    $10     NUMBER OF LANES EQUIPPED WITH DETECTOR(S) IN DIR. #1(N/B OR W/B) - MAX 8
00062          0011  A NLANE2 EQU    $11     NUMBER OF LANES EQUIPPED WITH DETECTOR(S) IN DIR. #2(S/B OR E/B) - MAX 8
00063          0012  A LDESW1 EQU    $12     UPSTREAM LANE DESCRIPTION BYTE IN DIRECTION #1.
00064                      *                 BIT 0 = 1 FOR LANE 1 DETECTOR ETC.IF 0 NO DETECTOR OR LANE EXISTS.
00065          0013  A LDESW2 EQU    $13     SAME AS LDESW1,BUT FOR DIRECTION #2.
00066          0014  A LDNW1  EQU    $14     DOWNSTREAM LANE DESCRIPTOR BYTE IN DIRECTION #1
00067          0015  A LDNW2  EQU    $15     SAME AS LDNW1,BUT FOR DIRECTION #2
00068          0016  A NVOL1  EQU    $16     NUMBER OF VOLUME COUNTS TO BE CALCULATED IN DIRECTION #1 - MAX 8
00069          0017  A NVOL2  EQU    $17     NUMBER OF VOLUME COUNTS TO BE CALCULATED IN DIRECTION #2 - MAX 8
00070                      *                 MAX TOTAL OF VOLUME COUNTS FOR BOTH DIRECTIONS EQUALS 8
00071          0018  A VOLDW1 EQU    $18     VOLUME COUNT DESCRIPTOR BYTE FOR VOLUME #1.
00072                      *                 LANE BIT=1 MEANS THAT THIS LANE CONTRIBUTES TO THIS VOLUME.
00073                      *                 LANES NUMBERED RIGHT TO LEFT IE BIT 0 = LANE 1(CENTRE).
00074                      *                 VOLUME DESCRIPTION STARTS WITH DIRECTION #1
00075                      *                 IMMEDIATELY FOLLOWED BY DIRECTION #2.
00076                      *                 THE UNUSED VOLUME DESCRIPTION WORDS WILL BE FILLED WITH BINARY ZEROES.
00077          0019  A VOLDW2 EQU    $19     VOLUME COUNT DESCRIPTOR BYTE FOR VOLUME #2
00078          001A  A VOLDW3 EQU    $1A     AS ABOVE FOR VOLUME #3
00079                      *
00080          001B  A VOLDW4 EQU    $1B     AS ABOVE FOR VOLUME #4
00081          001C  A VOLDW5 EQU    $1C     AS ABOVE FOR VOLUME #5
00082          001D  A VOLDW6 EQU    $1D     AS ABOVE FOR VOLUME #6
00083          001E  A VOLDW7 EQU    $1E     AS ABOVE FOR VOLUME #7
00084          001F  A VOLDW8 EQU    $1F     AS ABOVE FOR VOLUME #8
```

Figure 24/ Station Geometry and Data Definition Locations

```
0010   -- (display number of lanes in Direction 1)
E06    -- (there are 6 detector-equipped lanes)
A      -- (display number of lanes in Direction 2)
E05    -- (there are 5 detector-equipped lanes)
A      -- (display upstream description, Direction 1)
F      -- (go to hex/binary display)
0      -- (clear the data)
123456 -- (all lanes this direction have upstream detectors)
A      -- (advance to upstream descripton, Direction 2)
0      -- (clear the data)
12345  -- (only Lanes 1-5 have upstream detectors)
A      -- (advance to downstream description, Direction 1)
0      -- (clear the data)
12346  -- (all lanes this direction except 5 have downstream
           detector)
A      -- (advance to downstream description, Direction 2)
0      -- (clear the data)
12345  -- (Lanes 1-5 of this direction have downstream detectors)
9      -- (return to time mode)
```

6.5/ Define Volume Counts

Locations $0016-$001F (see Figure 24) must be initialized to define the lanes which are to be included in up to 8 different volume counts.

Suppose that in our hypothetical station we wish to accumulate the following data for Direction 1:
i/ number of vehicles using Lane 1
ii/ number of vehicles using Lanes 2, 3 and 4
iii/ number of vehicles using Lane 5

For Direction 2, we wish to accummulate:
i/ number of vehicles using Lanes 1 and 2
ii/ number of vehicles using Lanes 4 and 5

Note that we are not interested in data for Lane 3, Direction 2. The key sequence below would be used (from time mode) to define the above volume counts.

```
0016 -- (display number of volume counts for Direction 1)
E03  -- (we require 3 such counts)
```

```
A    -- (advance to volume counts for Direction 2)
E02  -- (we require 2 such counts)
A    -- (advance to volume count Descriptor 1)
F    -- (go to hex/binary display mode)
01   -- (clear byte, then set Lane 1)
A    -- (advance to volume count description 2)
0234 -- (clear byte, the set lanes 2, 3 and 4)
A    -- (advance to volume description 3)
05   -- (clear byte then set lane 5)
A    -- (advance to volume description 4)
012  -- (clear byte then set lanes 1 and 2)
A    -- (advance to volume description 5)
045  -- (clear byte then set lanes 4 and 5)
A0   -- (clear rest of volume descriptions)
A0   --
A0   --
9    -- (return to time mode)
```

At present, the unused volume descriptor bytes must be zeroed out. Future application releases may do this automatically.

6.6/ Detector Spacing and Width

In order to calculate vehicular speed and length, both the distance between loops (measure from center to center) and the width of the upstream loops must be known. These measurements must be stored (in units of 1/8 ft.) in the locations indicated in Figure 25. To compute the value to store, take the distance in feet, multiply by 8 and convert to hexadecimal.

The standard detector has a width of 6 feet and detectors are usually placed 26 feet apart. Suppose all of the loops in our sample station conform to this, except Lane 4 of Direction 1, where the detector loop separation is only 18 feet. In our example, all upstream detector widths are 6 feet, so we perform the following calculations:

6 feet x 8 = 48
convert to hex: 48 = $30

LOOP SEPARATION DIR. No. 1	LOOP SEPARATION DIR. No. 2	LOOP WIDTH DIR. No. 1	LOOP WIDTH DIR. No. 2	
0204	0205	0206	0207	LANE No. 1
0214	0215	0216	0217	LANE No. 2
0224	0225	0226	0227	LANE No. 3
0234	0235	0236	0237	LANE No. 4
0244	0245	0246	0247	LANE No. 5
0254	0255	0256	0257	LANE No. 6
0264	0265	0266	0267	LANE No. 7
0274	0275	0276	0277	LANE No. 8

Figure 25/ Detector Spacing and Width Locations

We enter the loop dimensions by keying in the sequence below (from time mode).

0206 -- (direction 1 loop widths)
E30 -- (Lane 1 upstream width)
CE30 -- (Lane 2 upstream width)
CE30 -- (Lane 3 upstream width)
CE30 -- (Lane 4 upstream width)
CE30 -- (Lane 5 upstream width)
CE30 -- (Lane 6 upstream width)
9 -- (return to time mode)
0207 -- (Direction 2 loop widths)
E30 -- (Lane 1 loop width)
CE30 -- (Lane 2 loop width)
CE30 -- (Lane 3 loop width)
CE30 -- (Lane 4 loop width)
CE30 -- (Lane 5 loop width)
9 -- (return to time mode)

Note that it is not necessary to zero out values for unused lanes.

Next, all loop separations (except Lane 4, Direction 1) are 26 feet.
26 feet x 8 = 208
convert to hex: 208 = $D0

For Lane 4, Direction 1:
18 feet x 8 = 144
convert to hex: 144 = $90

Enter these values (from time mode) with the following key sequence:
0204 -- (Direction 1 loop separation)
ED0 -- (Lane 1)
CED0 -- (Lane 2)
CED0 -- (Lane 3)
CE90 -- (Lane 4)
C -- (skip Lane 5, only 1 loop)
CED0 -- (Lane 6)
9 -- (return to time mode)

0205 -- (Direction 2 loop separation)
EDO -- (Lane 1)
CEDO -- (Lane 2)
CEDO -- (Lane 3)
CEDO -- (Lane 4)
CEDO -- (Lane 5)
9 -- (return to time mode)

Again, note that no values are entered for unused lanes.

6.7/ Master Initialization

Finally, a master initialization should be performed. To do this, the following key sequence is used from time mode:

006DE01 -- (enter 01 at location 006D)

Within 1 second, the value displayed should revert to 00, indicating that the initialization is done.

Note -- Although the previous descriptions have frequently used the 9 key to return to time mode, this is done only to enter a new address directly. This is unnecessary if the next address to be changed can be accessed conveniently through the stepping keys A, B, C and D.

7/ PROGRAM DESCRIPTIONS

7.1/ Overview of Program Structure

This section describes the overall structure of the PCS controller's system software (application software is mentioned but not described in detail). The PROM layout of this software is shown in Figure 26.

There are basically two types of routines that control the operation of the counting station. The first type includes those routines that are executed periodically at pre-defined time intervals (every 1/60 sec, every 1/30 sec, and so on). The second type are those routines executed to process interrupts. The boundary between these types is not distinct since some routines are shared by both the interrupt-driven programs and those that are not interrupt-driven. Refer to Figures 27 and 28 for a pictorial overview of program structure.

7.1.1/ Routines Not Driven by Interrupts

When the power to the controller is turned on, the routine POWRON is invoked to initialize the controller. SWI is invoked to update the software clock (once for every minute the controller was turned off, as indicated by a down-time accumulator). CALPRO is invoked to perform a complete PROM test, and then RAMCHK is called to test all of the controller's RAM. After that, control is passed to the executive routine EXEC.

EXEC controls the execution of all routines that are not driven by interrupts. Control is maintained by EXEC as long as the controller is turned on (except for the interrupt-driven routines). EXEC periodically executes system (and application) routines at predefined time frequencies. The time periods for the routines are indicated in Figure 27. EXEC uses a simple loop that examines time flags (set by the routine CLOCK), executes all the routines associated with the time flags, and then repeats the entire process. This is done as long as the controller is turned on, although occassionally the process is suspended temporarily by a clock or communication interrupt. The routines called by EXEC, the frequencies with which they are called, and their purposes are all described below.

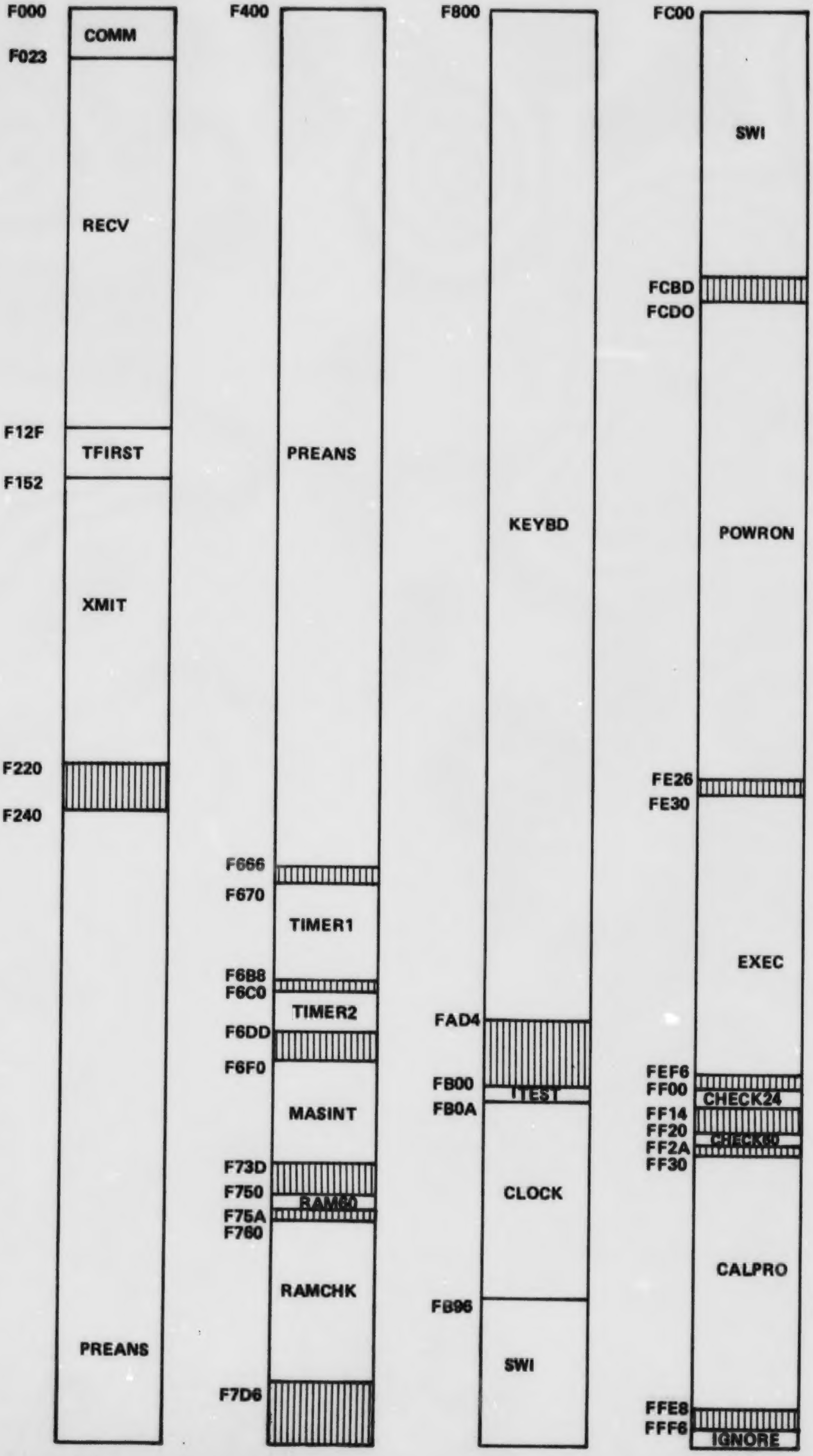

F000
COMM
F023
RECV
F12F
TFIRST
F152
XMIT
F220
F240
PREANS
F400
PREANS
F666
F670
TIMER1
F6B8
F6C0
TIMER2
F6DD
F6F0
MASINT
F73D
F750
RAM60
F75A
F760
RAMCHK
F7D6
F800
KEYBD
FAD4
FB00
ITEST
FB0A
CLOCK
FB96
SWI
FC00
SWI
FCBD
FCD0
POWRON
FE26
FE30
EXEC
FEF6
FF00
CHECK24
FF14
FF20
CHECK60
FF2A
FF30
CALPRO
FFE8
FFF6
IGNORE

Figure 26/ PROM Layout

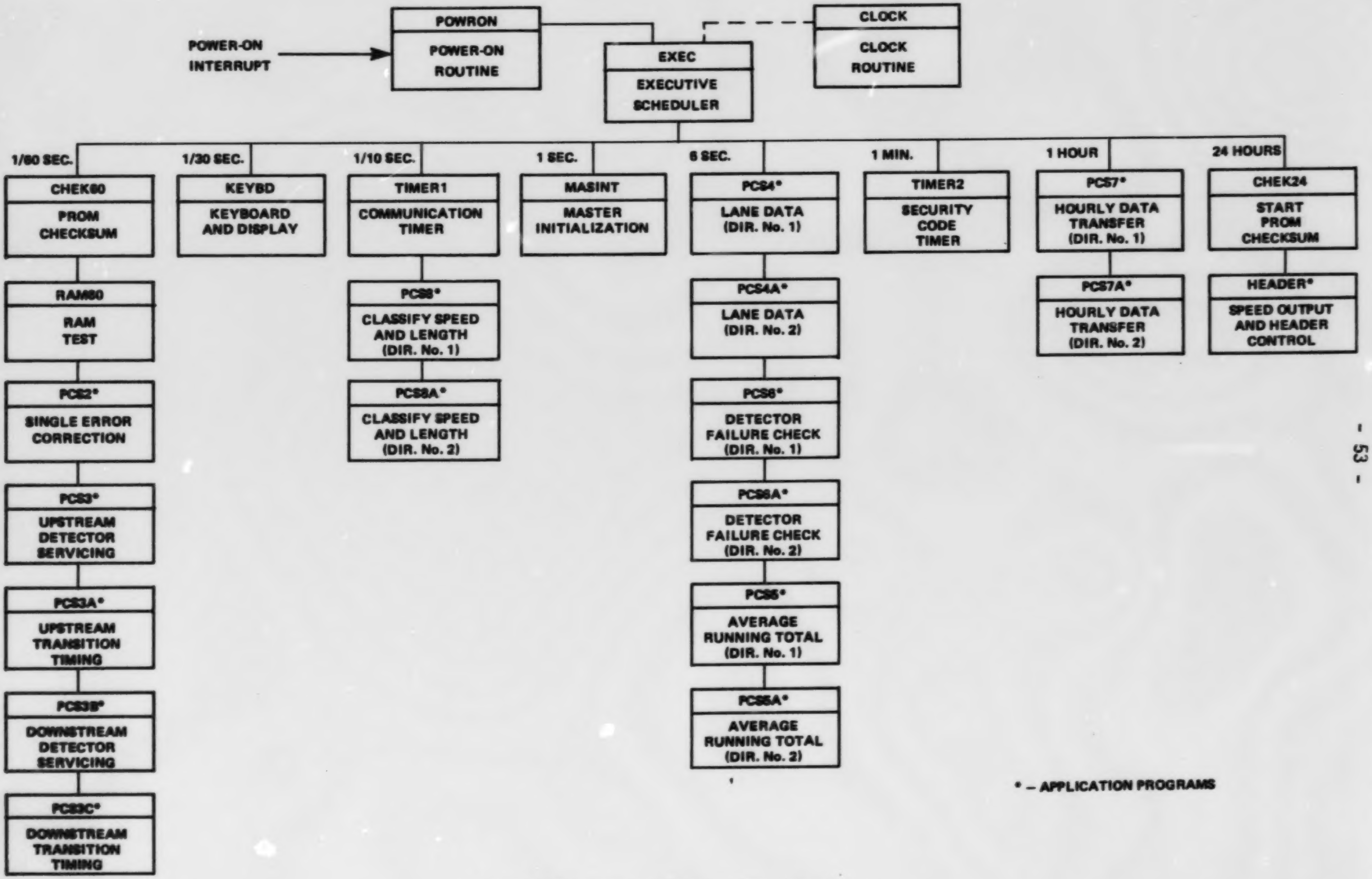

POWER-ON INTERRUPT
POWRON
POWER-ON ROUTINE
EXEC
EXECUTIVE SCHEDULER
CLOCK
CLOCK ROUTINE
1/60 SEC.
CHEK60
PROM CHECKSUM
RAM60
RAM TEST
PCS2*
SINGLE ERROR CORRECTION
PCS3*
UPSTREAM DETECTOR SERVICING
PCS3A*
UPSTREAM TRANSITION TIMING
PCS3B*
DOWNSTREAM DETECTOR SERVICING
PCS3C*
DOWNSTREAM TRANSITION TIMING
1/30 SEC.
KEYBD
KEYBOARD AND DISPLAY
1/10 SEC.
TIMER1
COMMUNICATION TIMER
PCS8*
CLASSIFY SPEED AND LENGTH (DIR. No. 1)
PCS8A*
CLASSIFY SPEED AND LENGTH (DIR. No. 2)
1 SEC.
MASINT
MASTER INITIALIZATION
6 SEC.
PCS4*
LANE DATA (DIR. No. 1)
PCS4A*
LANE DATA (DIR. No. 2)
PCS6*
DETECTOR FAILURE CHECK (DIR. No. 1)
PCS6A*
DETECTOR FAILURE CHECK (DIR. No. 2)
PCS5*
AVERAGE RUNNING TOTAL (DIR. No. 1)
PCS5A*
AVERAGE RUNNING TOTAL (DIR. No. 2)
1 MIN.
TIMER2
SECURITY CODE TIMER
1 HOUR
PCS7*
HOURLY DATA TRANSFER (DIR. No. 1)
PCS7A*
HOURLY DATA TRANSFER (DIR. No. 2)
24 HOURS
CHEK24
START PROM CHECKSUM
HEADER*
SPEED OUTPUT AND HEADER CONTROL
* – APPLICATION PROGRAMS

Figure 27/ Overview of Program Execution

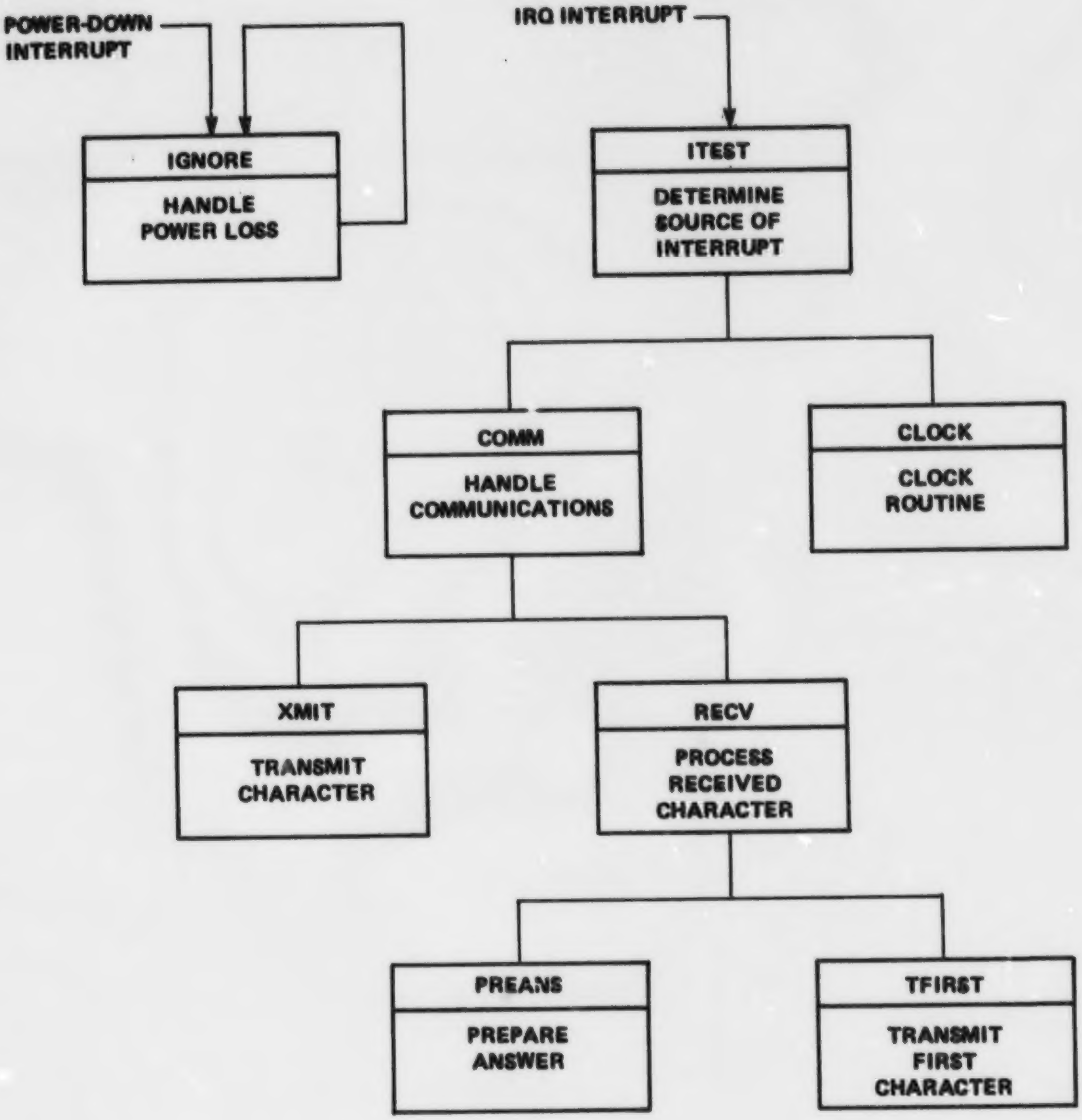

POWER-DOWN INTERRUPT
IRQ INTERRUPT
IGNORE
HANDLE POWER LOSS
ITEST
DETERMINE SOURCE OF INTERRUPT
COMM
HANDLE COMMUNICATIONS
CLOCK
CLOCK ROUTINE
XMIT
TRANSMIT CHARACTER
RECV
PROCESS RECEIVED CHARACTER
PREANS
PREPARE ANSWER
TFIRST
TRANSMIT FIRST CHARACTER

Figure 28/ Overview of Interrupt-driven Entry Points

CHEK60 (called every 1/60 sec)
This routine invokes CALPRO to test PROM, unless such a test has already been completed that day. Only 1 byte is processed every 1/60 sec.

RAM60 (called every 1/60 sec)
This routine invokes RAMCHK to test 1 byte of RAM, unless a RAM test has already been completed that day or unless the PROM test is still being performed. Once the PROM test is complete, the RAM test is started.

KEYBD (called every 1/30 sec)
This routine processes any keys pressed at the controller's keyboard and displays information on the controller's display.

TIMER1 (called every 1/10 sec)
This is the communication timer routine. It is used to detect the loss of part of a communication message during transmission or receipt of such a message. If the loss occurred during receipt of a message, a response message is initiated by invoking TFIRST, which sends the first character of the message.

MASINT (called every 1 sec)
This routine will initialize system and application variables (the latter is accomplished by invoking INIT) if such an initialization has been requested.

TIMER2 (called every 1 min)
This is the security code timer routine and it simply clears the security code (to prohibit RAM changes) 255 minutes after a communication message from the central station has changed the controller's RAM.

CHECK24 (called every 24 hours)
This routine, executed at the end of every day, initiates a PROM checksum test (which, when finished, will in turn initiate a RAM test).

Application Programs (called at various time intervals)
These routines handle the collection of traffic data. Their precise operation is not discussed in this document.

7.1.2/ Routines Driven by Interrupts

The routine IGNORE handles the non-maskable interrupt (NMI) which occurs when the power to the controller is removed. The routine does nothing, it just provides an orderly means of shutting down the controller.

All IRQ interrupts cause ITEST to be invoked. ITEST determines the source of the interrupt and takes appropriate action. If the interrupt was caused by the communication interface, routine COMM is invoked. Otherwise, the interrupt must have come from the real-time clock (every 1/60 sec), so control is passed to CLOCK.

COMM invokes XMIT if the interrupt was caused by a successful character transmission. XMIT then transmits the next character of the message (if there is anything left to send). COMM invokes RECV if the interrupt was caused by the successful receipt of a character. RECV processes the character and once a logically correct message has been received, PREANS is invoked to process the message and prepare a response. Once the entire message has been received, TFIRST is invoked to transmit the first character of the response message prepared by PREANS.

CLOCK updates the time by 1/60 second, setting the time flags to be used by EXEC. If 1 minute has been reached, SWI is entered to update the time by 1 minute.

7.2/ Power-Up Routine (POWRON)

Purpose

The routine POWRON is executed every time the power to the controller is turned on. It performs the following functions.

- Find the end of contiguous RAM and locate the stack at its end.
- Initialize communications.
- Check date and time variables, correct if necessary.
- Save date and time of power failure.
- Enable interrupts.
- Initialize keyboard and other variables.
- Correct time for power loss.
- Check integrity of PROM and perform RAM test.

At the end of POWRON, control is given to the routine EXEC.

Description

First, POWRON finds the end of continuous RAM memory. It does this by writing to memory at successively higher addresses until the value read back is not the same as the value written (which means the location is at an empty socket). At the end of RAM, 32 bytes are reserved for the system stack and the highest available user RAM address is stored for future use in the variable MRAM.

Next, POWRON initializes communication hardware and enters a receive state, in which it is prepared to accept communication messages from the central computer.

Next, time and date variables are checked to ensure they are valid. For example, the seconds counter must be a valid BCD quantity not exceeding 59. These checks are performed only to ensure valid time variables exist after the very first power-up (when RAM values are unpredictable). After this first power-up, the CLOCK subroutine and auxiliary power supply (batteries) will maintain the validity of time variables. Note that time variables are not protected against deliberate RAM changes (although they will be checked on subsequent power-ups). If any single time variable is found to be invalid, they are all reset to zeros. The date (year, month and day) and time (hours and minutes) of the power failure are saved for further reference.

After that, interrupts are enabled, permitting clock and communication interrupts.

The keyboard is then reset and the display timer is started.

Next, a down-time counter is incremented and a power failure alarm is set. There is a special alarm set if the power failure is longer than 255 minutes. In such a case the time variables are no longer reliable.

Then, the time is updated to correct for the power loss:

New time = Old time + Time that power was down
+ Average loss of down-time accumulator

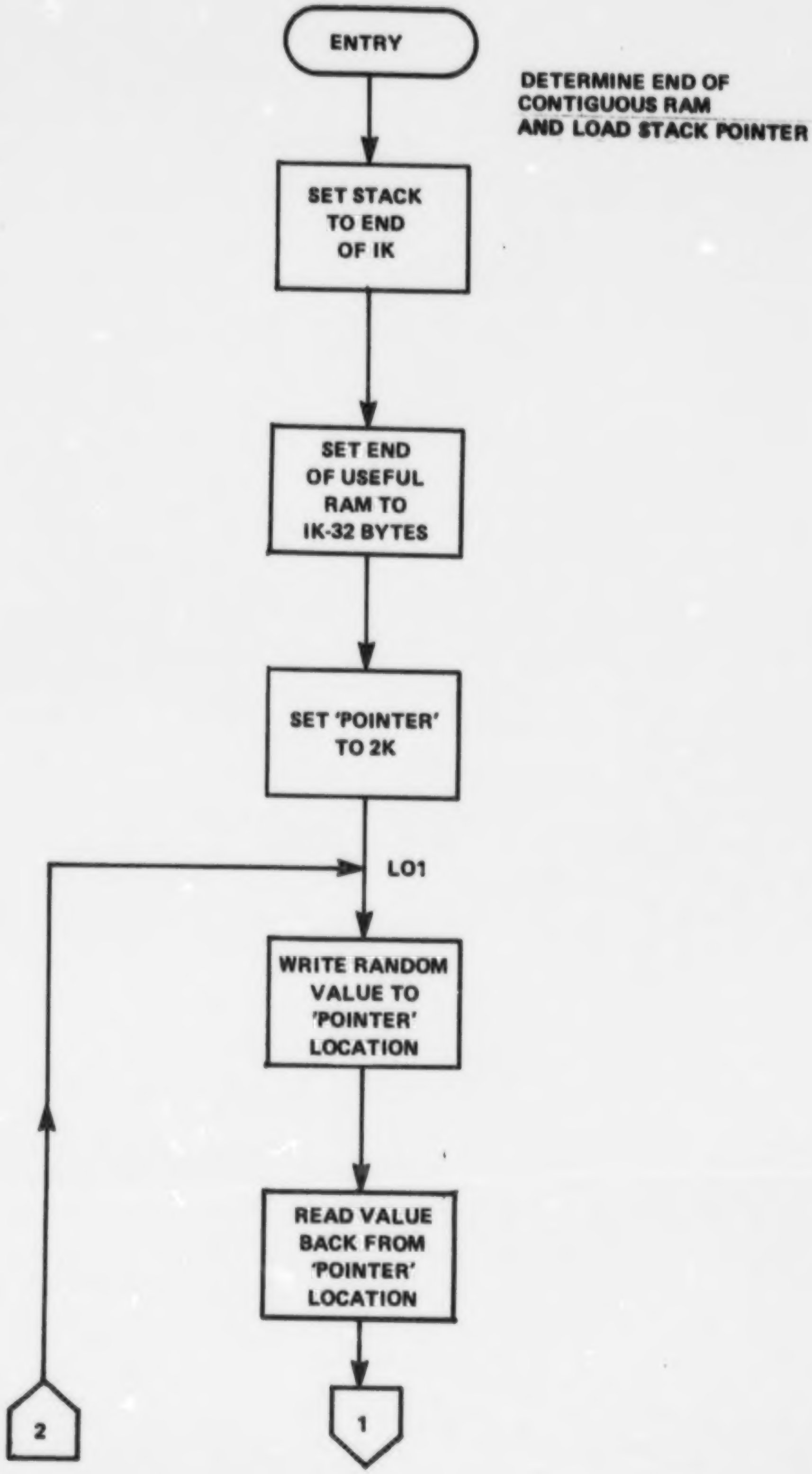
Following that, a PROM checksum is performed to ensure program integrity and a RAM test is performed. Ordinarily these checks proceed at 1 byte every 1/60 sec. On power-up, however, the entire check is done at once. This takes well over 1 second to do, but since clock interrupts have been enabled the clock can update the time as the memory checks proceed.

At the end of the PROM checksum, performed by subroutine CALPRO, a RAM test is initialized. The RAM test is then performed (by POWRON) by invoking subroutine RAMCHK. With 20K bytes of RAM memory in the controller, this test can take about 8 seconds.

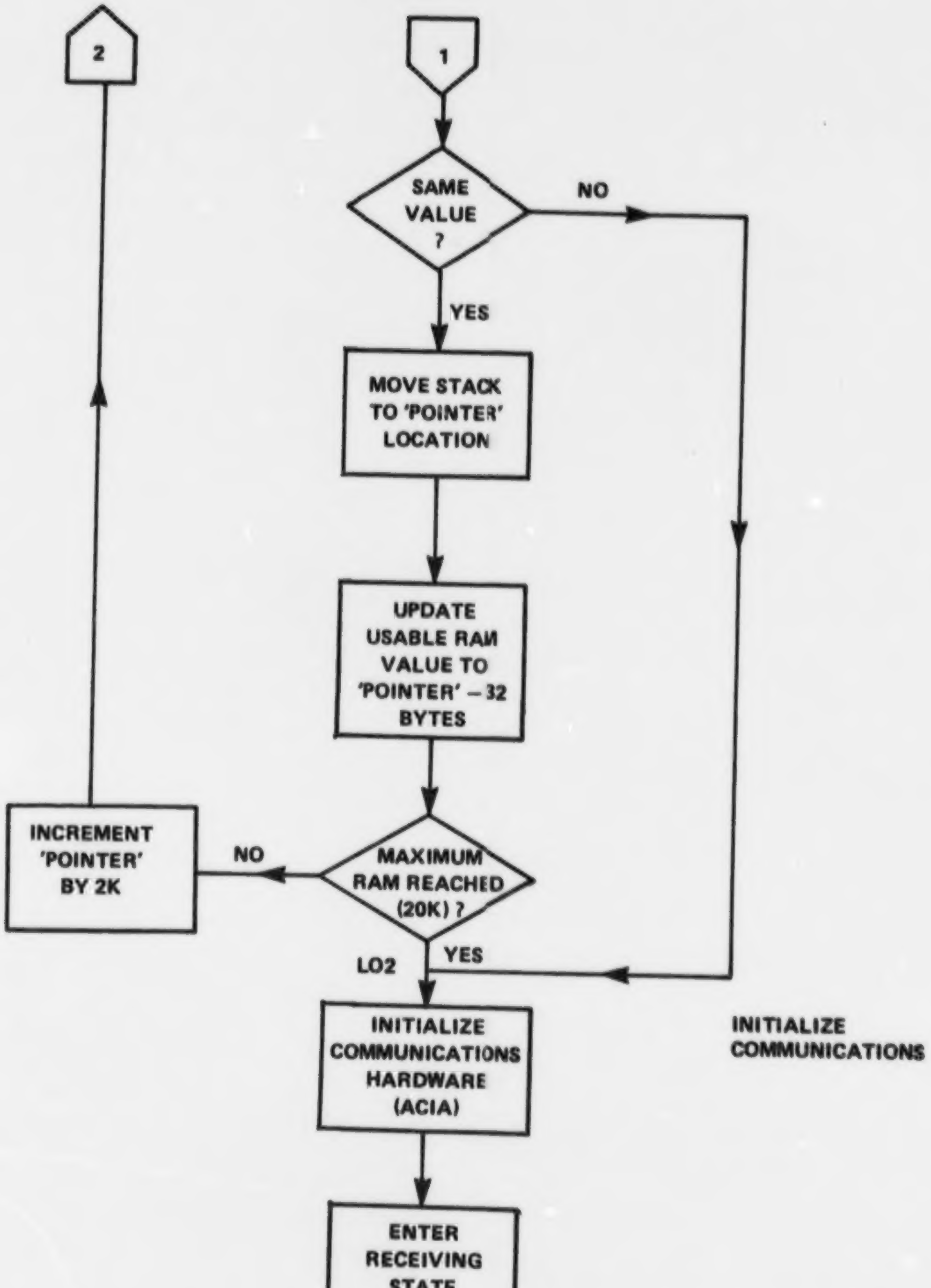
Control is then passed to the routine EXEC.

POWER UP ROUTINE (POWRON)

Page 1 of 8

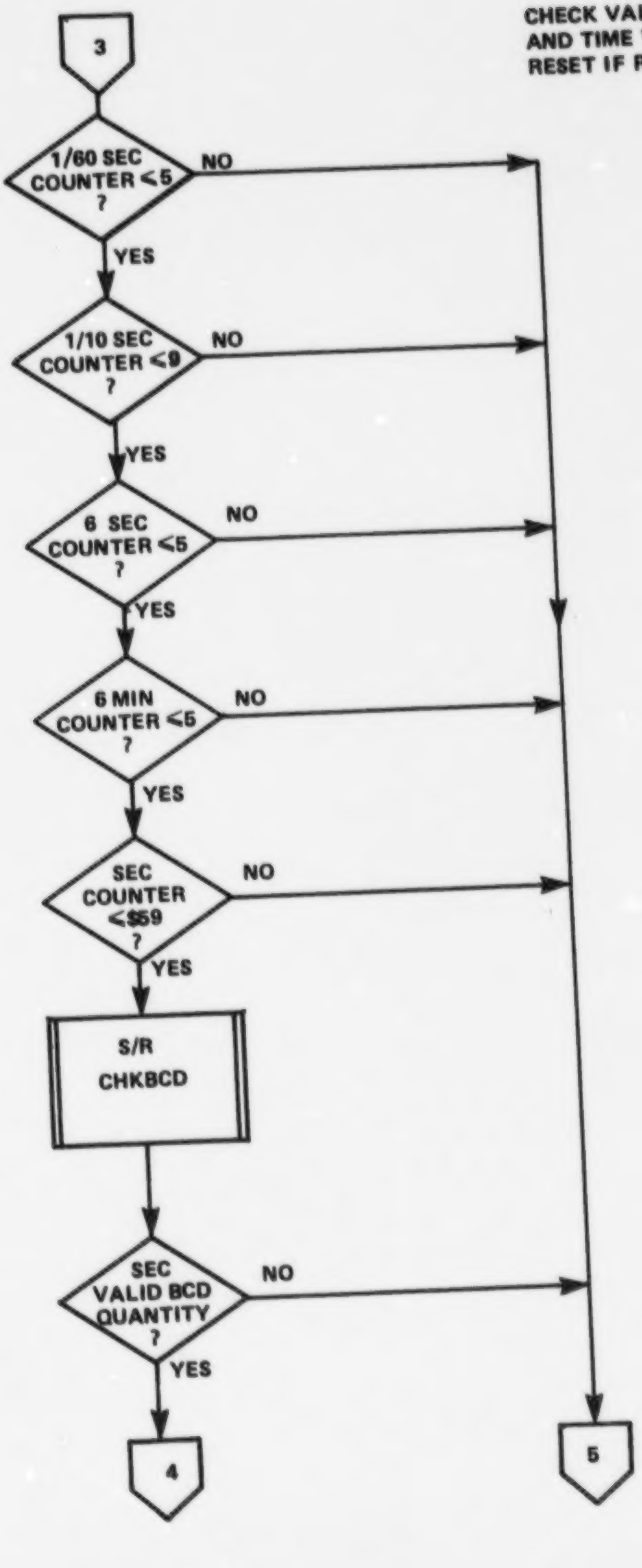

ENTRY
DETERMINE END OF CONTIGUOUS RAM AND LOAD STACK POINTER
SET STACK TO END OF 1K
SET END OF USEFUL RAM TO 1K-32 BYTES
SET 'POINTER' TO 2K
LO1
WRITE RANDOM VALUE TO 'POINTER' LOCATION
READ VALUE BACK FROM 'POINTER' LOCATION
2
1

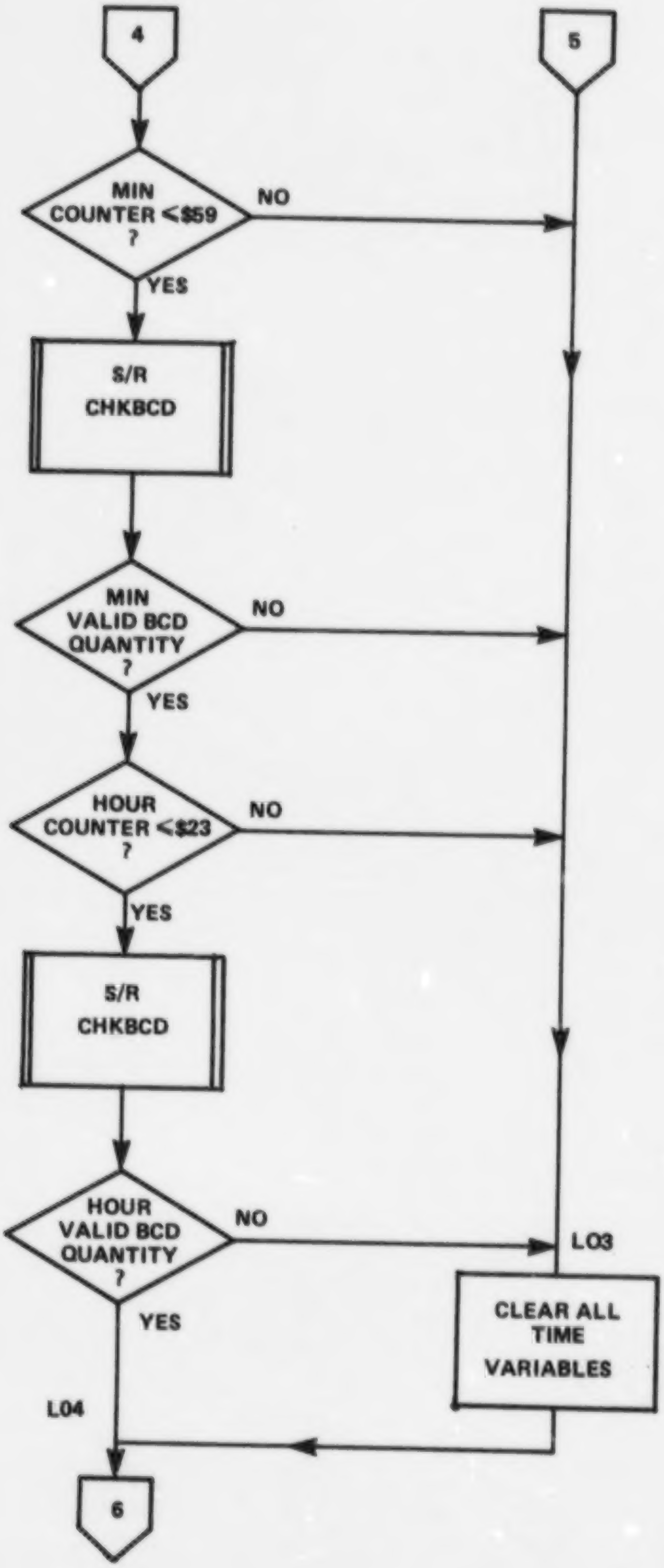
POWER UP ROUTINE (POWRON) Page 2 of 8

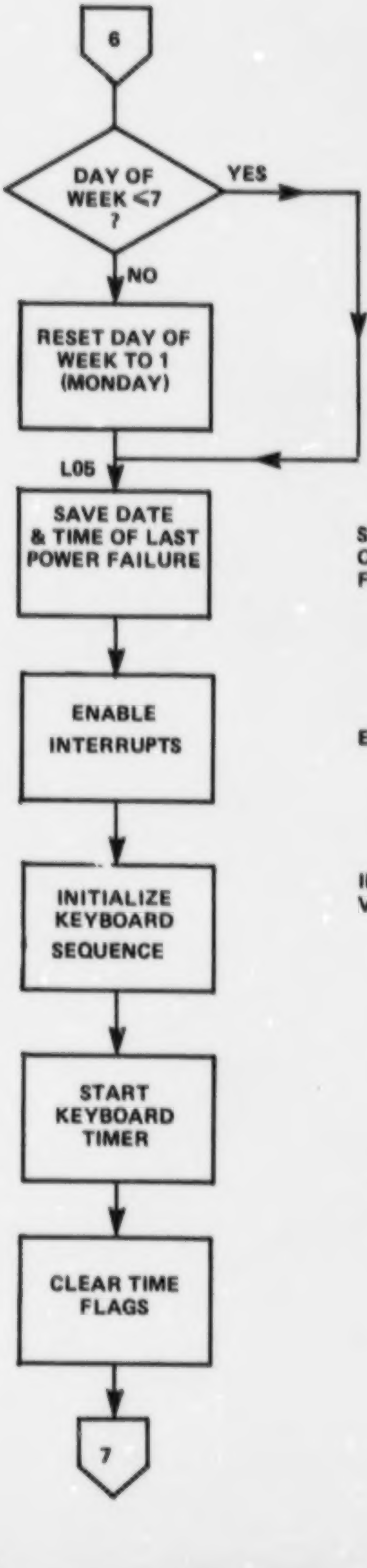

2
1
SAME VALUE ?
NO
YES
MOVE STACK TO 'POINTER' LOCATION
UPDATE USABLE RAM VALUE TO 'POINTER' – 32 BYTES
INCREMENT 'POINTER' BY 2K
NO
MAXIMUM RAM REACHED (20K) ?
YES
LO2
INITIALIZE COMMUNICATIONS HARDWARE (ACIA)
INITIALIZE COMMUNICATIONS
ENTER RECEIVING STATE
3

POWER UP ROUTINE (POWRON)

CHECK VALIDITY OF DATE
AND TIME VARIABLES –
RESET IF REQUIRED


3
1/60 SEC COUNTER ≤5 ?
NO
YES
1/10 SEC COUNTER ≤9 ?
NO
YES
6 SEC COUNTER ≤5 ?
NO
YES
6 MIN COUNTER ≤5 ?
NO
YES
SEC COUNTER ≤$59 ?
NO
YES
S/R CHKBCD
SEC VALID BCD QUANTITY ?
NO
YES
4
5

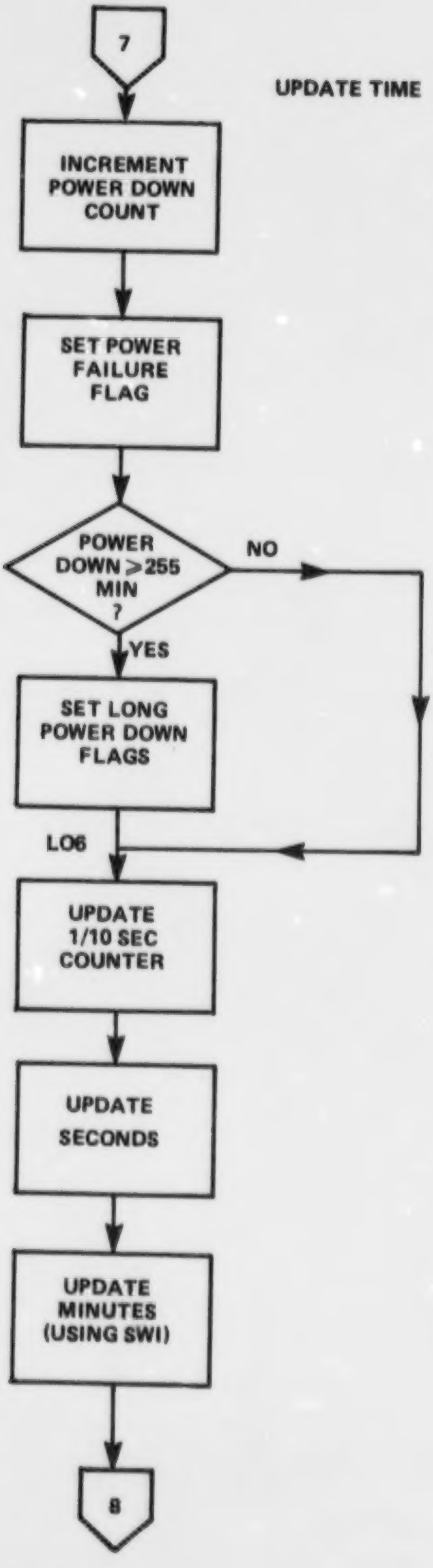

4
5
MIN COUNTER <$59 ?
NO
YES
S/R CHKBCD
MIN VALID BCD QUANTITY ?
NO
YES
HOUR COUNTER <$23 ?
NO
YES
S/R CHKBCD
HOUR VALID BCD QUANTITY ?
NO
YES
LO3
CLEAR ALL TIME VARIABLES
LO4
6

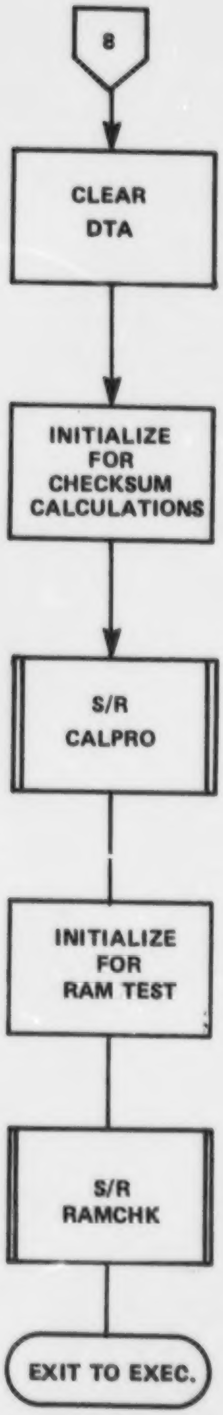

6
DAY OF WEEK <7 ?
YES
NO
RESET DAY OF WEEK TO 1 (MONDAY)
L05
SAVE DATE & TIME OF LAST POWER FAILURE
ENABLE INTERRUPTS
INITIALIZE KEYBOARD SEQUENCE
START KEYBOARD TIMER
CLEAR TIME FLAGS
7

SAVE DATE & TIME OF LAST POWER FAILURE

ENABLE INTERRUPTS

INITIALIZE OTHER VARIABLES

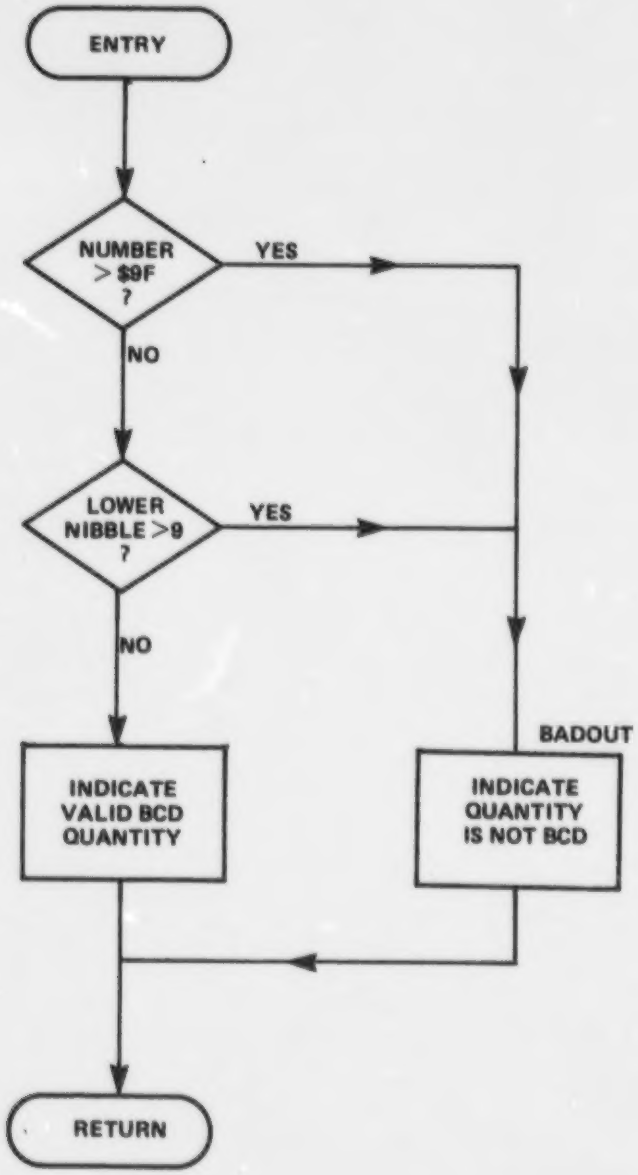

7
UPDATE TIME
INCREMENT POWER DOWN COUNT
SET POWER FAILURE FLAG
POWER DOWN ≥ 255 MIN ?
NO
YES
SET LONG POWER DOWN FLAGS
LO6
UPDATE 1/10 SEC COUNTER
UPDATE SECONDS
UPDATE MINUTES (USING SWI)
8


8
CLEAR DTA
INITIALIZE FOR CHECKSUM CALCULATIONS
S/R CALPRO
INITIALIZE FOR RAM TEST
S/R RAMCHK
EXIT TO EXEC.

PROM CHECKSUM CALCULATIONS

RAM TEST

CHECK FOR A VALID BCD NUMBER SUBROUTINE (CHKBCD)


ENTRY
NUMBER > $9F ?
YES
NO
LOWER NIBBLE > 9 ?
YES
NO
BADOUT
INDICATE VALID BCD QUANTITY
INDICATE QUANTITY IS NOT BCD
RETURN

7.3/ Executive Routine (EXEC)

Purpose

This routine calls all programs which are not interrupt-driven. Programs are executed on a time basis using flags TMFLG1 and TMFLG2 set by the clock routine (CLOCK). The following time intervals are available:

1/60 sec, 1/30 sec, 1/20 sec, 1/10 sec, 1/2 sec, 1 sec, 6 sec, 30 sec,
1 min, 6 min, 15 min, 20 min, 30 min,
1 hour, and 24 hours.

For example, every 1/10 seconds the routine EXEC will call all programs that are supposed to be executed every 1/10 second.

Description

EXEC examines the 16 time flags in TMFLG1 and TMFLG2 in increasing order from the 1/60 sec flag to the 24-hour flag. Note that there are actually two 24-hour flags, the second one being reserved for future addition of a time flag.

If there are no flags set yet, EXEC goes into a wait state to wait for an interrupt. After the interrupt the time flags are checked. If all the flags are zero, the wait state is entered again. Otherwise, the routine EXEC causes execution of other routines as outlined below.

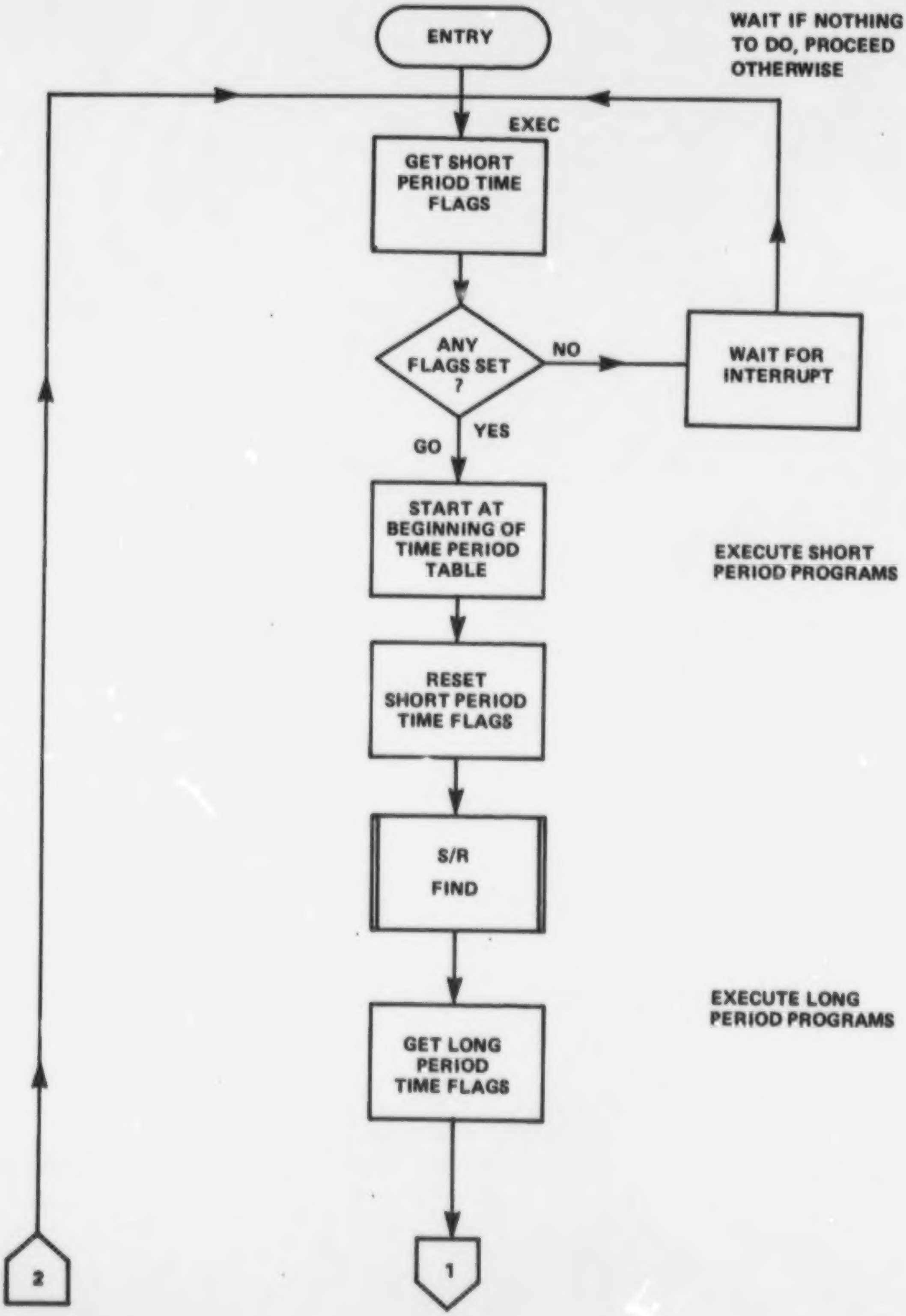
The routines to be executed are accessed through the table EXTABL. Each entry in this table points to a sequential list of subroutines to execute. Each such list is terminated by $FFFF.

Each of the flags from 1/60 sec to 24 hours is examined in turn. If it is set, the programs for that time period are executed. This is done by two calls to subroutine FIND (one call for the short time flags TMFLG1 and one call for the long time flags TMFLG2). After all programs belonging to the time flags that were set have been executed, the routine EXEC goes back to the wait state, waiting for an interrupt.

Note: -- To add a new program it is necessary only to change the subroutine lists, adding any new routine name to the appropriate list. This routine must finish with an RTS instruction, and the programmer should keep in mind the time required to execute the routine and its frequency of execution.

Refer to the program listing for a list of those routines called by EXEC and when they are executed (i.e., how frequently).

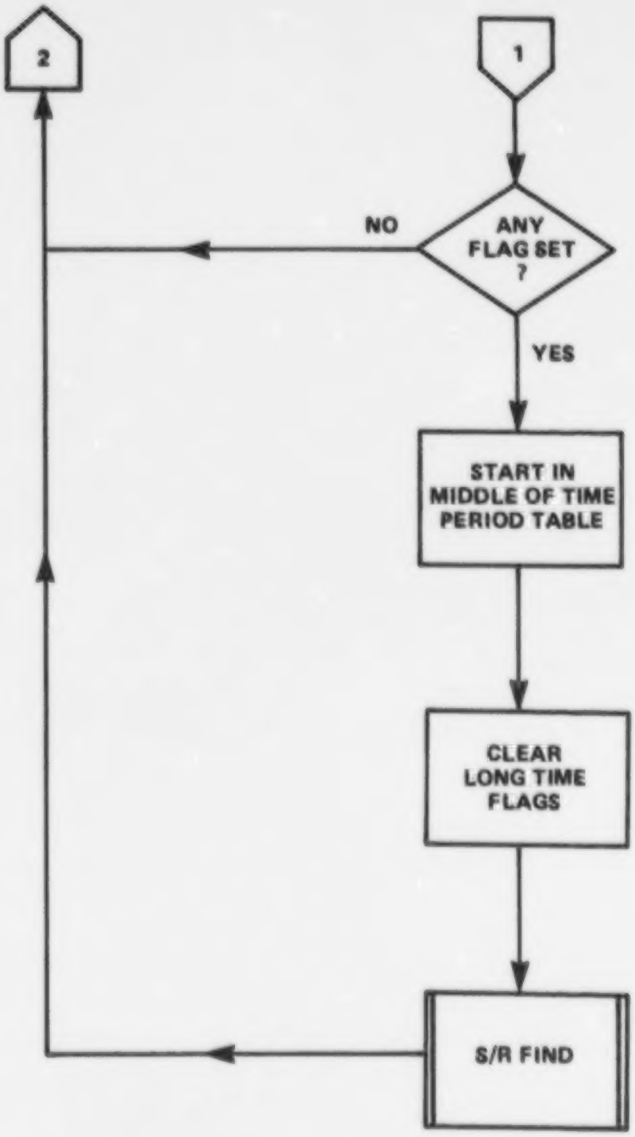
EXECUTIVE ROUTINE (EXEC) Page 1 of 4


ENTRY
EXEC
GET SHORT PERIOD TIME FLAGS
ANY FLAGS SET ?
NO
WAIT FOR INTERRUPT
YES
GO
START AT BEGINNING OF TIME PERIOD TABLE
RESET SHORT PERIOD TIME FLAGS
S/R FIND
GET LONG PERIOD TIME FLAGS
1
2

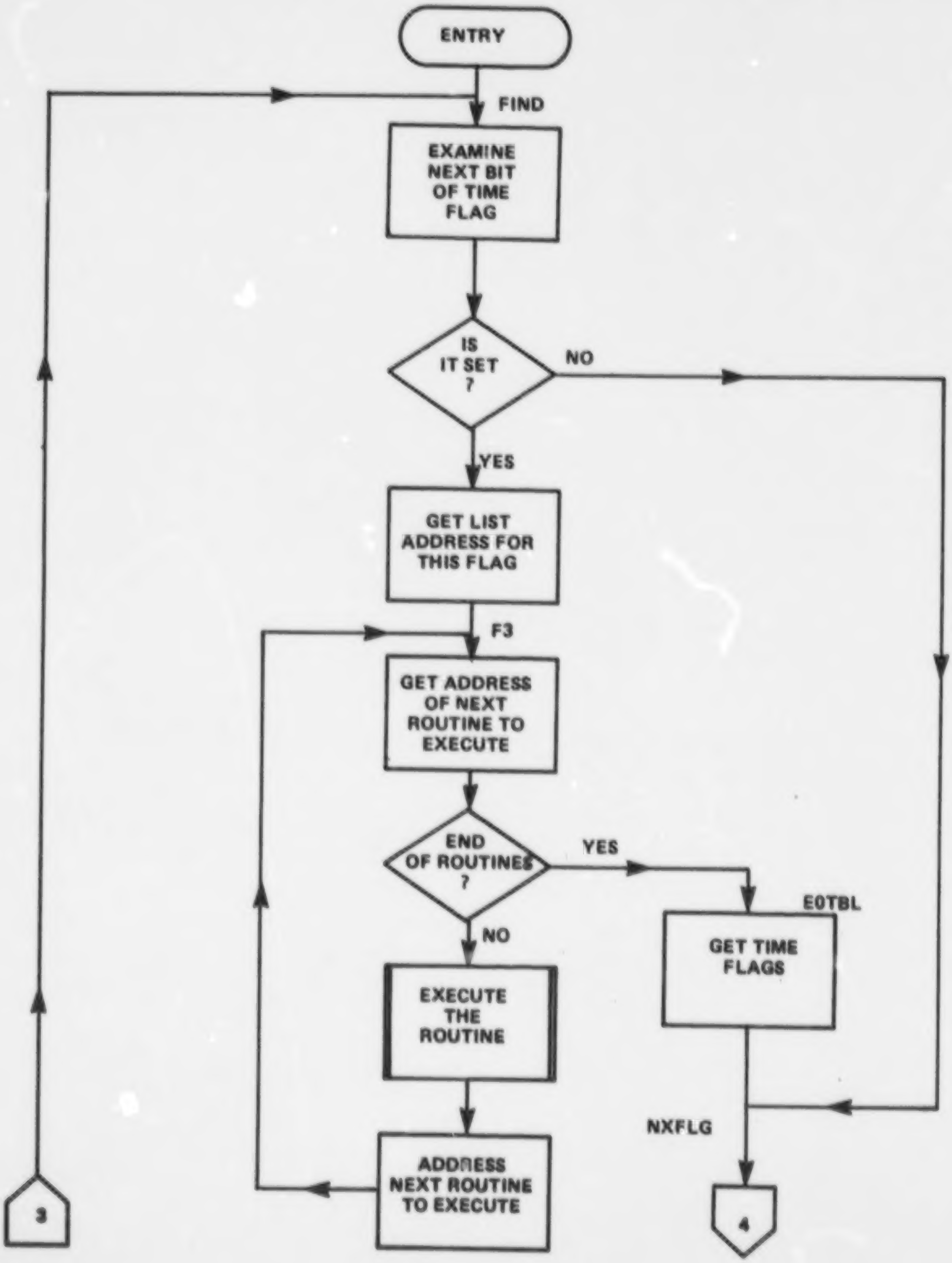
WAIT IF NOTHING TO DO, PROCEED OTHERWISE

EXECUTE SHORT PERIOD PROGRAMS

EXECUTE LONG PERIOD PROGRAMS

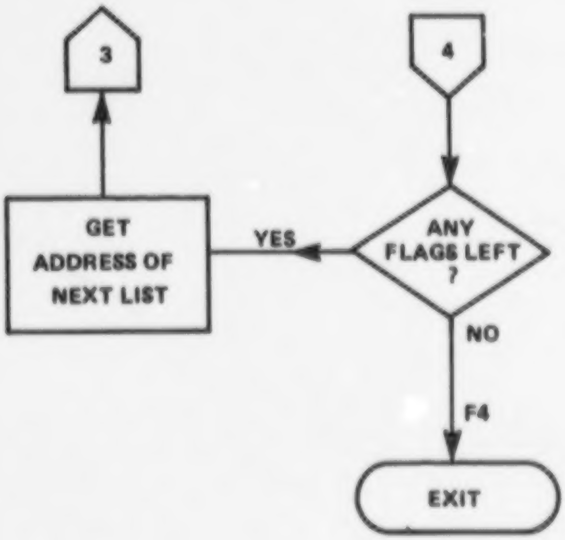

2
1
ANY FLAG SET ?
NO
YES
START IN MIDDLE OF TIME PERIOD TABLE
CLEAR LONG TIME FLAGS
S/R FIND

EXECUTIVE ROUTINE (EXEC)
EXECUTE PROGRAMS SUBROUTINE (FIND)


ENTRY
FIND
EXAMINE NEXT BIT OF TIME FLAG
IS IT SET ?
NO
YES
GET LIST ADDRESS FOR THIS FLAG
F3
GET ADDRESS OF NEXT ROUTINE TO EXECUTE
END OF ROUTINES ?
YES
NO
EOTBL
GET TIME FLAGS
EXECUTE THE ROUTINE
NXFLG
ADDRESS NEXT ROUTINE TO EXECUTE
3
4

EXECUTIVE ROUTINE (EXEC)
EXECUTE PROGRAMS SUBROUTINE (FIND)

Page 4 of 4


3
4
GET ADDRESS OF NEXT LIST
YES
ANY FLAGS LEFT ?
NO
F4
EXIT

7.4/ Clock Routine (CLOCK)

Purpose

This routine is entered whenever an interrupt is sent to the microprocessor (assuming, of course, that interrupts are enabled). It determines the source of the interrupt and control is then transferred to the appropriate routine to process the interrupt. There are two possible sources of interrupts:

- the communication interface (ACIA), and
- the 1/60 sec clock.

If it is a communication interrupt, control is passed to COMM. Otherwise, it must have been a clock interrupt. In this case, the time and date variables are updated as required. Also, the time flags TMFLG1 and TMFLG2 are set, and every 255 minutes the display is blanked out and the keyboard reset (unless a key has recently been pushed). All this processing is done every 1/60 second.

Description

This routine is entered at entry point ITEST. It tests for a communication interrupt and passes control to the communication handling routine COMM if required. Otherwise, the interrupt must have been caused by the real-time clock, and the processing outlined below is performed.

First, the real time clock is reset.

Next, the 60 Hz pulse count is incremented and the 60 Hz bit of TMFLG1 is set:

- If 2 pulses have occurred (2 x 1/60 = 1/30 sec), the 30 Hz bit of TMFLG1 is set.

- If 3 pulses have occurred (3 x 1/60 = 1/20 sec), the 20 Hz bit of TMFLG1 is set.

- If 6 pulses have occurred (6 x 1/60 = 1/10 sec), the 10 Hz and 20 Hz bits of TMFLG1 are set and the 60 Hz pulse count is reset to zero to start a new cycle -- also, the 1/10 sec counter is incremented -- if 6 pulses have not yet occurred the routine terminates.

After 5/10 sec has passed the 1/2 sec bit of TMFLG1 is set and the routine terminates.

After 10/10 sec (i.e., 1 second) has passed, both the 1/2 sec and 1 sec bits of TMFLG1 are set, the 1/10 sec counter is cleared, and the 1-second and 6-second counters are both incremented.

If it has not yet reached 1 second, the routine terminates.

After 6 seconds have passed, the 6 sec bit of TMFLG1 is set, the 6 sec counter is cleared, and the day of week pointer (for the display lights) is set. After 30 seconds have elapsed, the 30 sec bit of TMFLG1 is set and the display timer is incremented. If the timer reaches 255 minutes, the display is blanked out, the security code is cleared, and the keyboard sequence is reset (the display timer is reset at every keypush). The routine then terminates.

After 1 minute has passed, the 30 sec bit of TMFLG1 and the 1 min bit of TMFLG2 are set and the seconds counter is cleared.

If it has not yet reached 1 minute, the routine terminates.

The program now reaches the entry point SWI to update the time by 1 minute (this entry point is used by the routine POWRON at power-up time).

The 6 sec counter is cleared and the minute counter is incremented:
- after 20 min or 40 min, the 20 min bit of TMFLG2 is set;
- after 15 min or 45 min, the 15 min bit of TMFLG2 is set;
- after 30 min, the 15 min and 30 min bits of TM[illegible]G2 are set.

The 6 min counter is incremented.
After 6 minutes have passed, the 6 min bit of TMFLG2 is set and the 6 min counter is cleared.

After 1 hour, the routine CLOCK does the following:
- sets 6 min, 15 min, 20 min, 30 min and 1 hour bits of TMFLG2;

- clears 1 min and 6 min counters;
- increments hours counter;
- at 2:00 am on the last Sunday of April or October, a daylight savings adjustment is made using subroutine DAYLIT (unless it has already been done today).

If 1 hour has not yet passed, the routine terminates.

After 24 hours, the 24 hour bits of TMFLG2 are set, the hour count is cleared, the day of week is changed (reset to Monday if necessary), and both the day of year and day of month are incremented.

If 24 hours have not yet passed the routine terminates.

At the end of the month (as determined by using subroutine NOFDAY), the day of month is set to 1 and the month is incremented.

If it is not yet the end of the month, the routine terminates.

At the end of the year, the day of year is set to 1, the month is set to 1, and the year is incremented.

The routine then terminates.

7.4.1/ Find the Number of Days in a Given Month Subroutine (NOFDAY)

Purpose

This subroutine (used by CLOCK) finds the number of days in a given month.

Description

The given month is a BCD number from 1 to 12. This is used to index into the table MONTBL, which contains the number of days (BCD) in each month. Since a BCD quantity is used as an index, there are 6 empty slots in the table, corresponding to hex values $0A to $0F.

If the month is February, then the year is examined. If it is a leap year, the number of days is increased by 1.

The number of days in the month is then stored in DIM, and the routine terminates.

7.4.2/ Daylight Savings Adjustment Subroutine (DAYLIT)

Purpose

This subroutine (used by CLOCK) moves the clock ahead 1 hour or back 1 hour if a daylight savings time adjustment is needed. It is called at 2:00 am every Sunday, except when an adjustment has already been made that day.

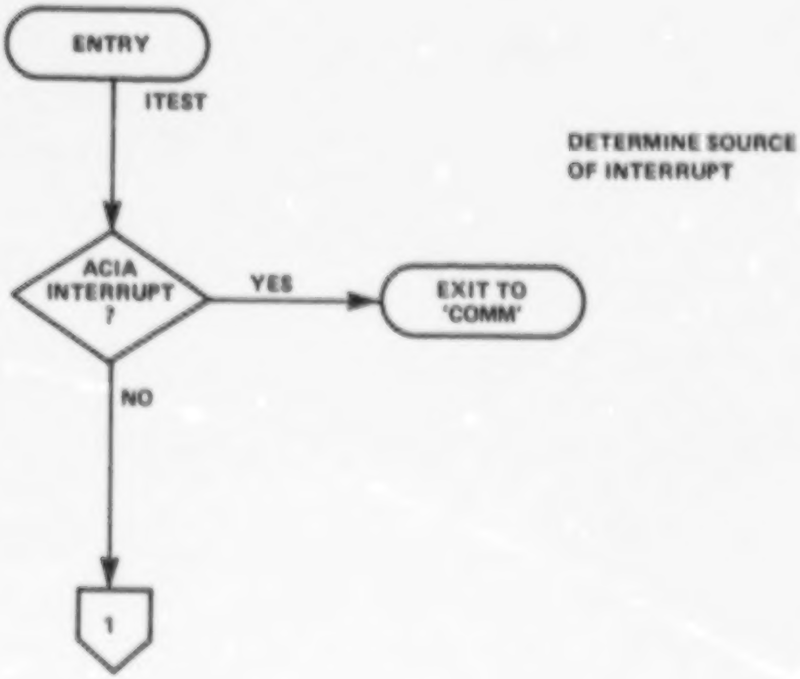
Description

The current day of the month is compared to the end of the month. If it is not the last Sunday of the month, DAYLIT terminates.

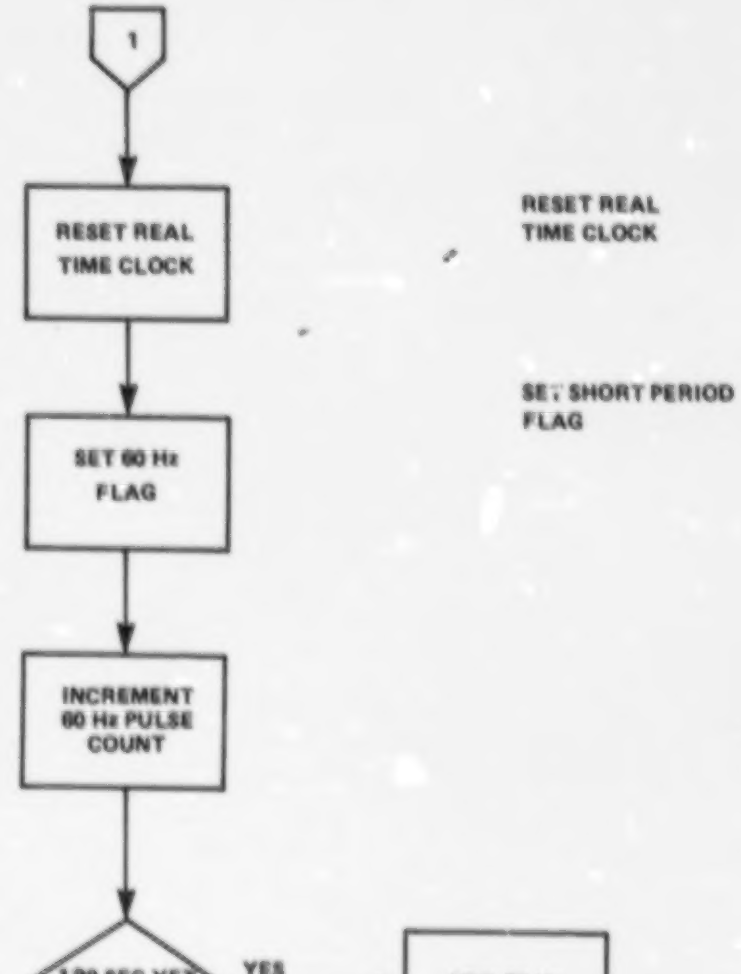
If it is the last Sunday, the month is examined.
If it is April, the clock is advanced 1 hour.
If it is October, the clock is set back 1 hour.

The subroutine then terminates.

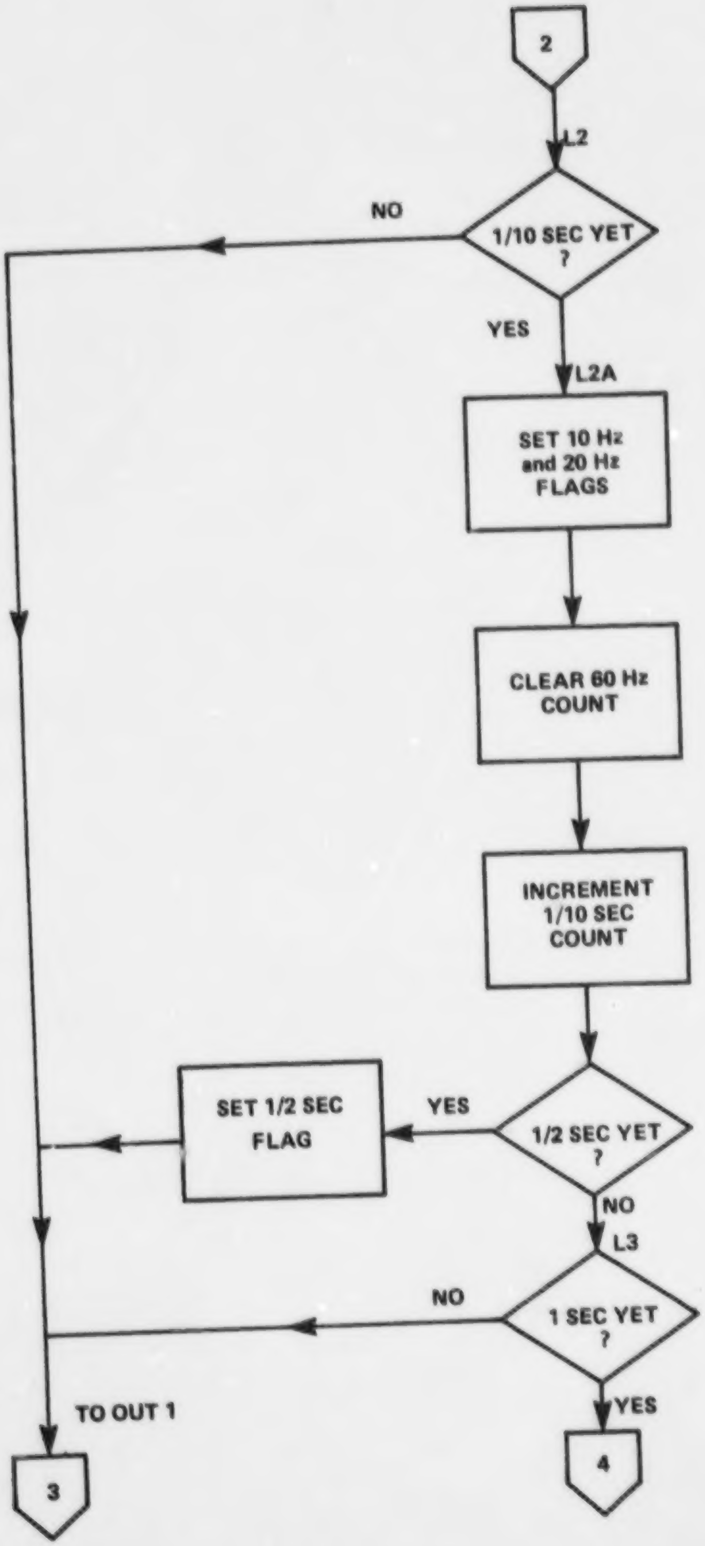
CLOCK ROUTINE (CLOCK)


ENTRY
ITEST
DETERMINE SOURCE OF INTERRUPT
ACIA INTERRUPT ?
YES
EXIT TO 'COMM'
NO
1

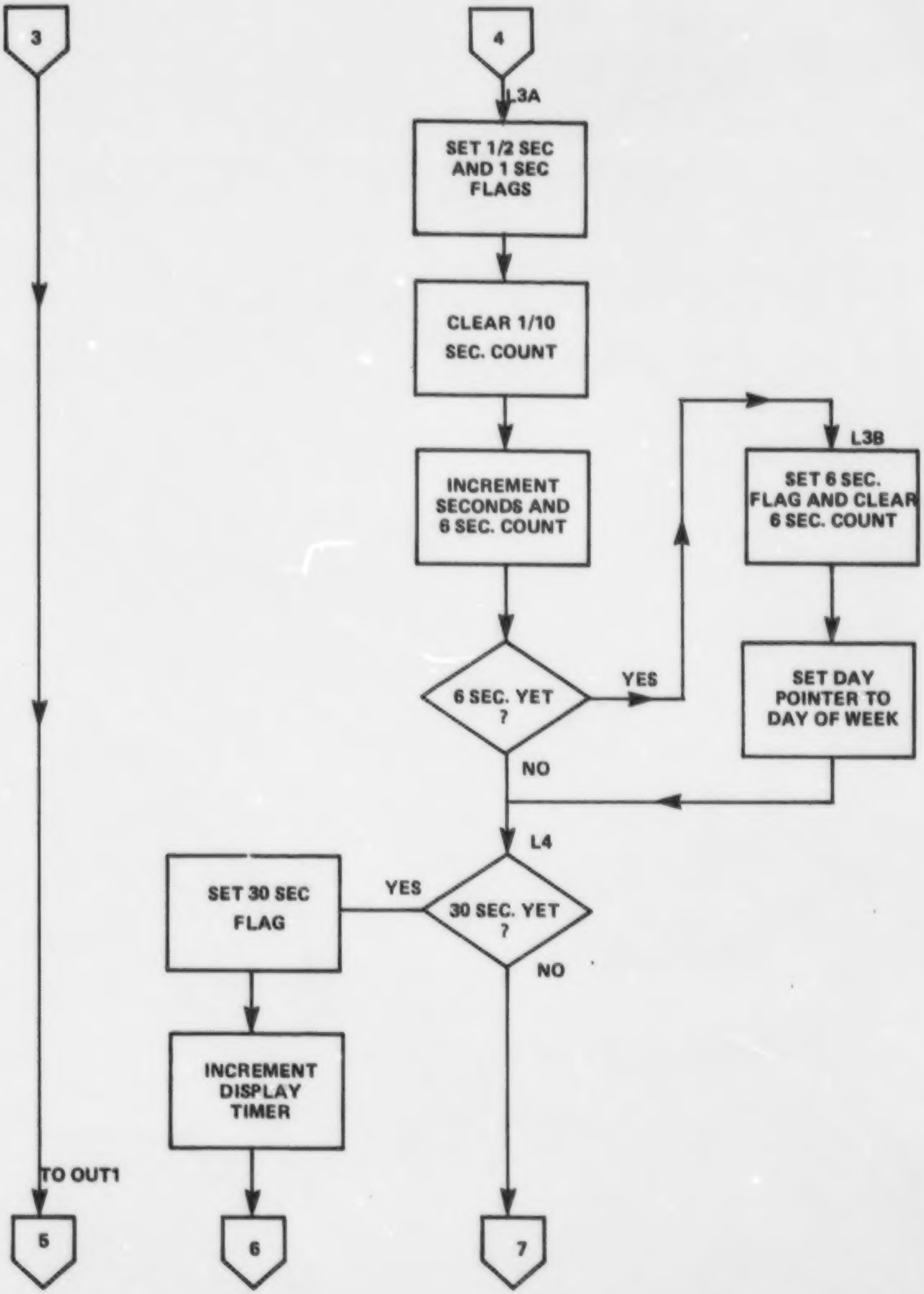
CLOCK ROUTINE (CLOCK)


1
RESET REAL TIME CLOCK
SET 60 Hz FLAG
INCREMENT 60 Hz PULSE COUNT
1/30 SEC YET ?
YES
SET 30 Hz FLAG
NO
L1
1/20 SEC YET ?
YES
SET 20 Hz FLAG
NO
2
RESET REAL TIME CLOCK
SET SHORT PERIOD FLAG

CLOCK ROUTINE (CLOCK)

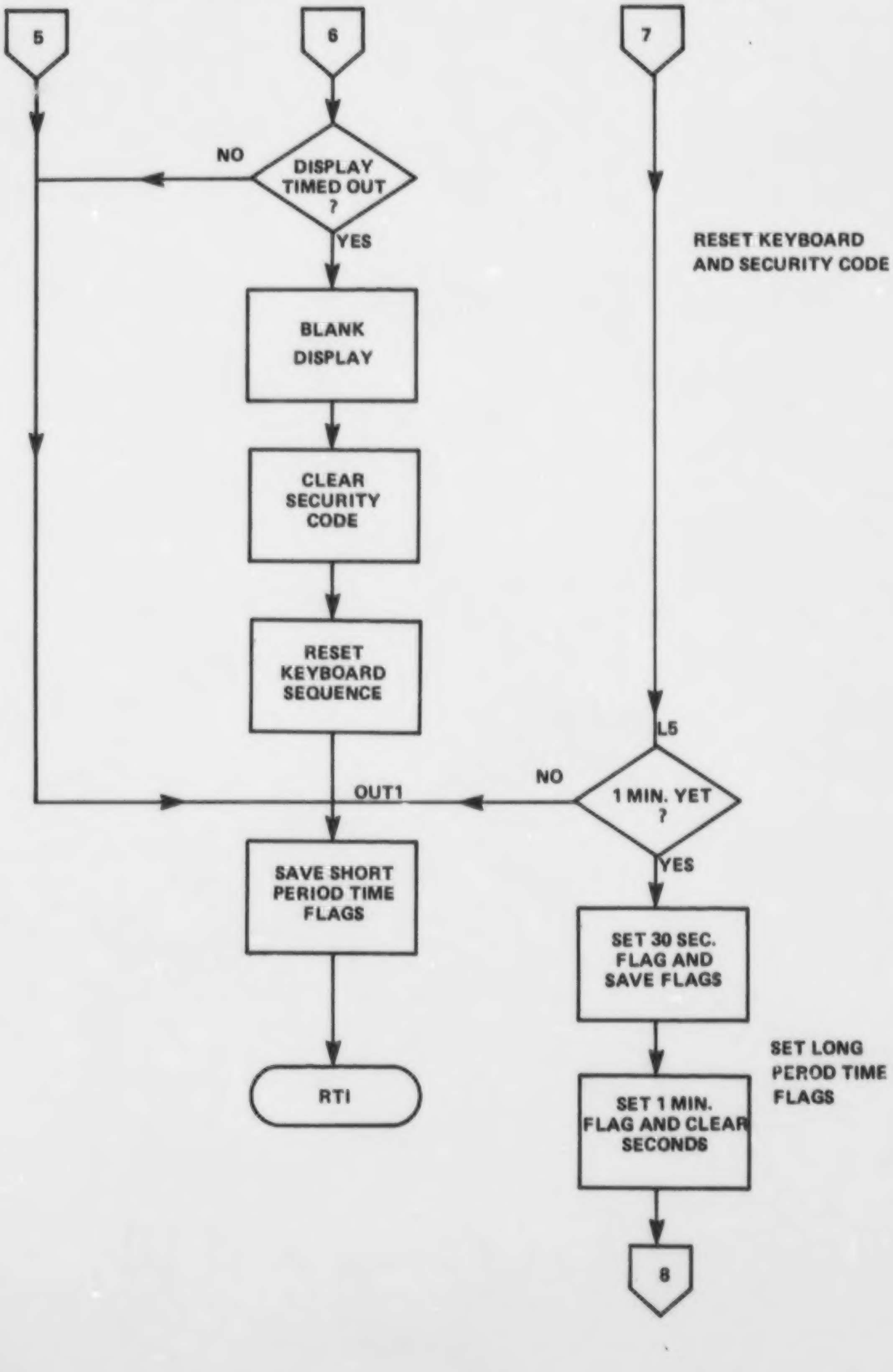

2
L2
NO
1/10 SEC YET ?
YES
L2A
SET 10 Hz and 20 Hz FLAGS
CLEAR 60 Hz COUNT
INCREMENT 1/10 SEC COUNT
SET 1/2 SEC FLAG
YES
1/2 SEC YET ?
NO
L3
NO
1 SEC YET ?
YES
TO OUT 1
3
4

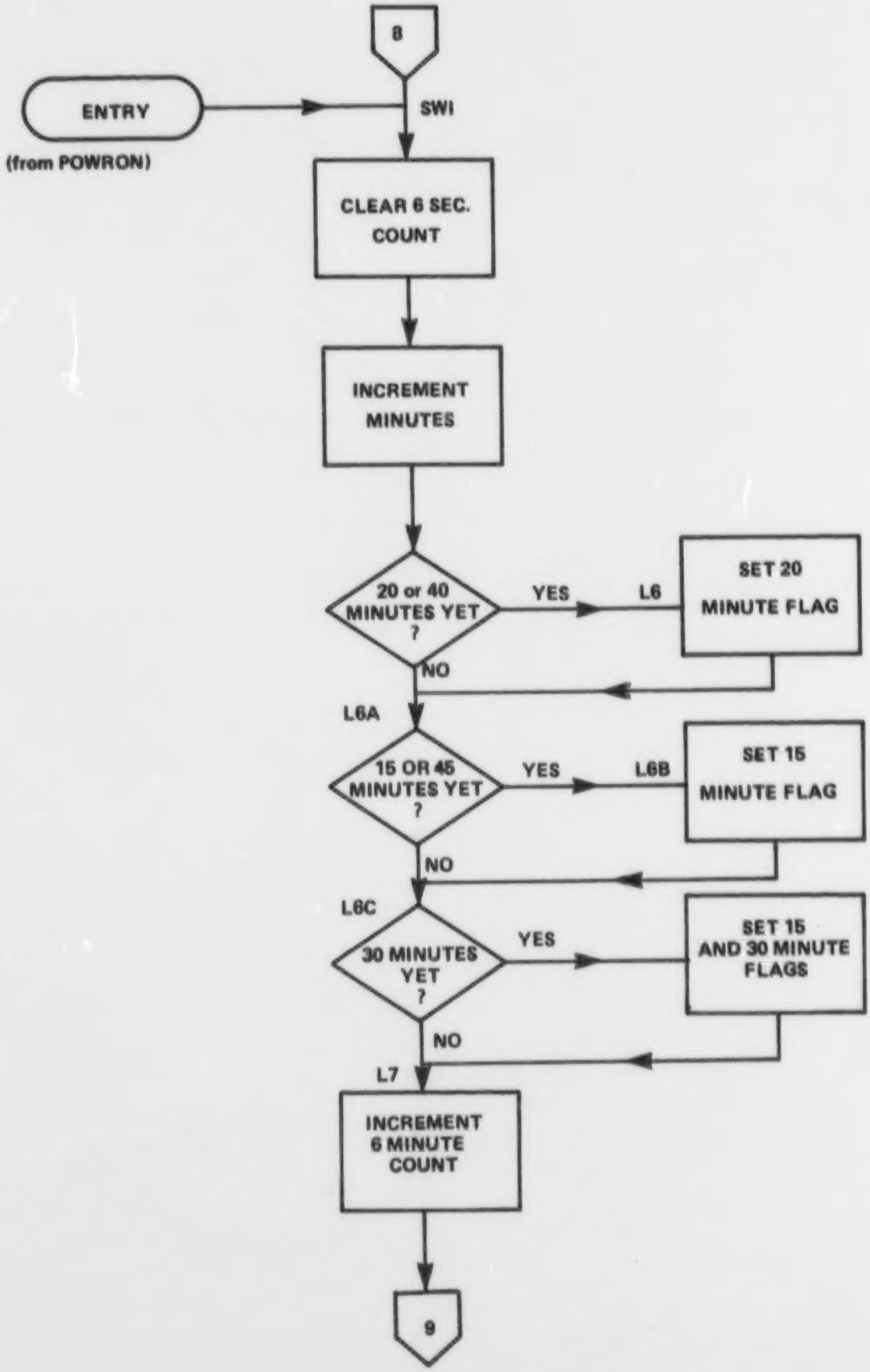
CLOCK ROUTINE (CLOCK)

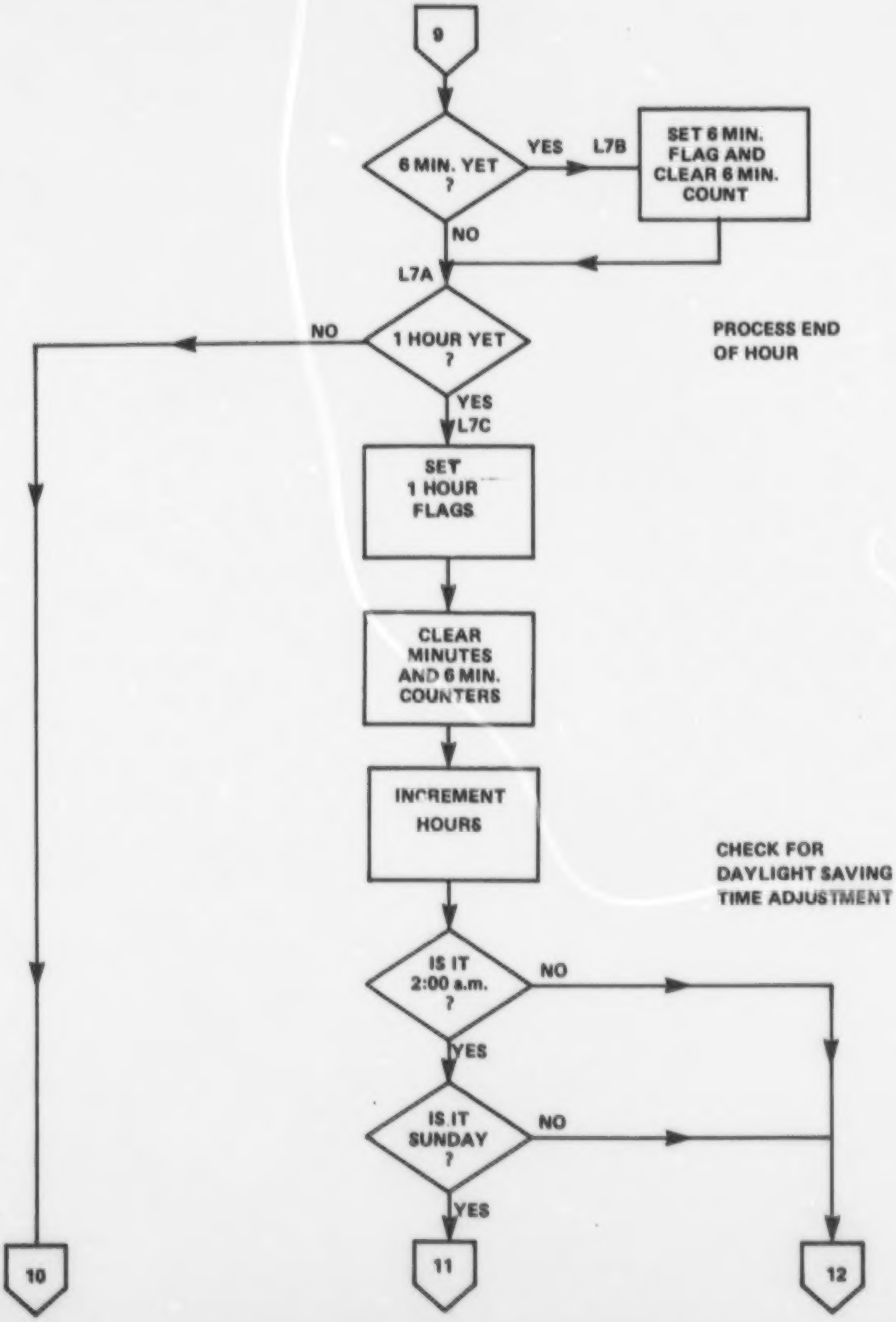

3
4
L3A
SET 1/2 SEC AND 1 SEC FLAGS
CLEAR 1/10 SEC. COUNT
INCREMENT SECONDS AND 6 SEC. COUNT
6 SEC. YET ?
YES
NO
L3B
SET 6 SEC. FLAG AND CLEAR 6 SEC. COUNT
SET DAY POINTER TO DAY OF WEEK
L4
30 SEC. YET ?
YES
NO
SET 30 SEC FLAG
INCREMENT DISPLAY TIMER
TO OUT1
5
6
7

CLOCK ROUTINE (CLOCK)

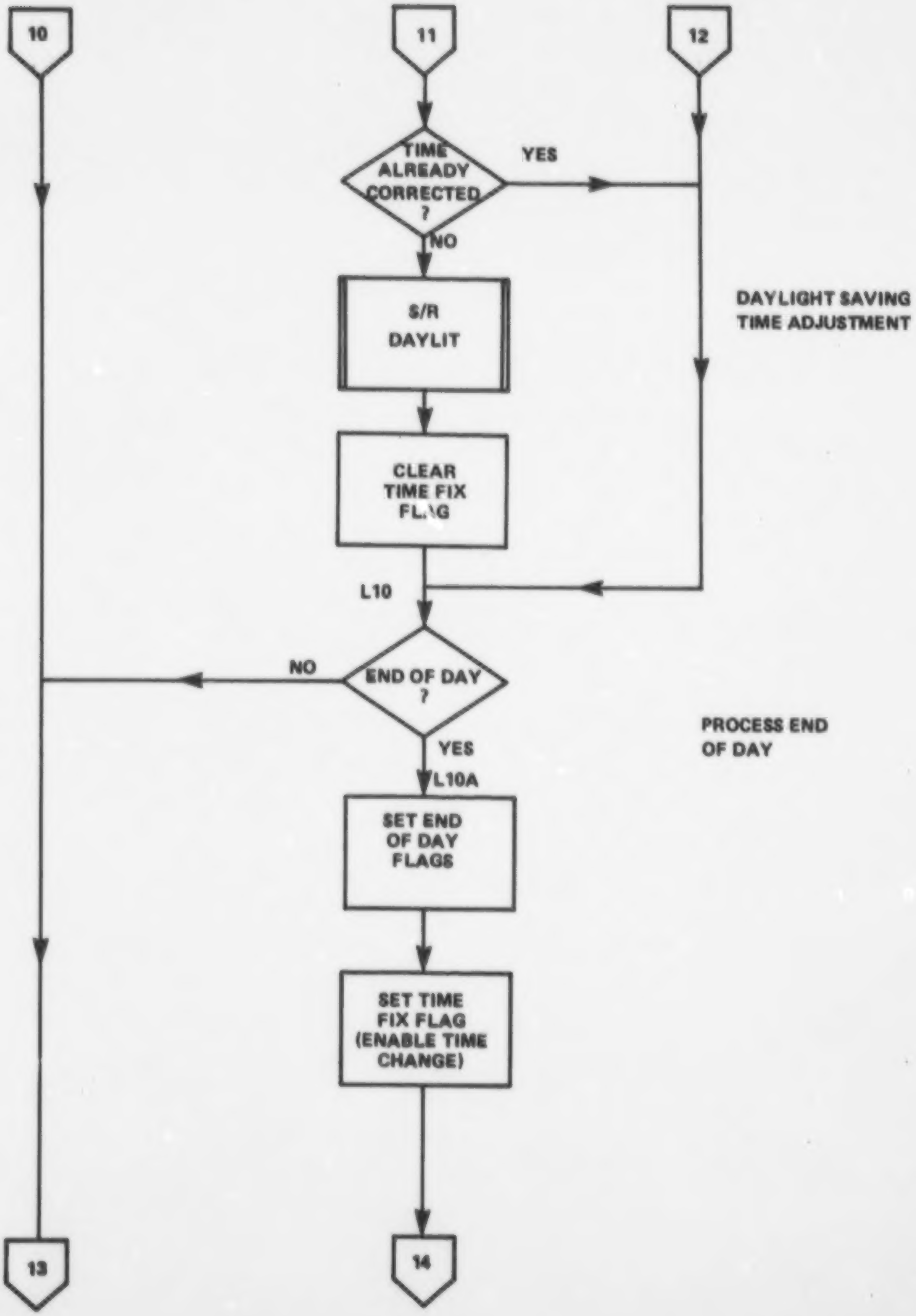
Page 5 of 12


5
6
7
NO
DISPLAY TIMED OUT ?
YES
RESET KEYBOARD AND SECURITY CODE
BLANK DISPLAY
CLEAR SECURITY CODE
RESET KEYBOARD SEQUENCE
L5
NO
OUT1
1 MIN. YET ?
YES
SAVE SHORT PERIOD TIME FLAGS
SET 30 SEC. FLAG AND SAVE FLAGS
SET LONG PEROD TIME FLAGS
RTI
SET 1 MIN. FLAG AND CLEAR SECONDS
8

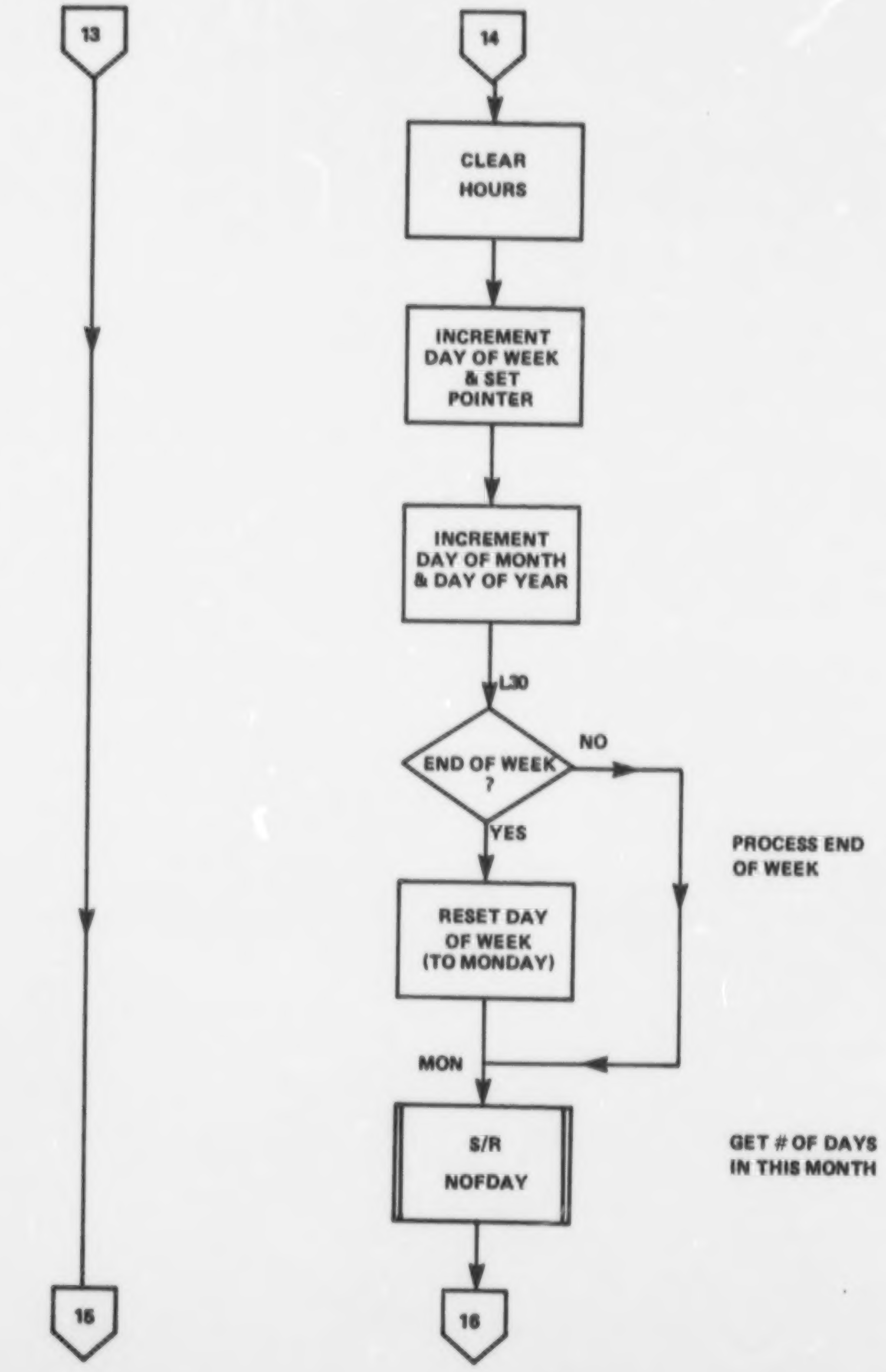

8
ENTRY
(from POWRON)
SWI
CLEAR 6 SEC. COUNT
INCREMENT MINUTES
20 or 40 MINUTES YET ?
YES
L6
SET 20 MINUTE FLAG
NO
L6A
15 OR 45 MINUTES YET ?
YES
L6B
SET 15 MINUTE FLAG
NO
L6C
30 MINUTES YET ?
YES
SET 15 AND 30 MINUTE FLAGS
NO
L7
INCREMENT 6 MINUTE COUNT
9


9
6 MIN. YET ?
YES
L7B
SET 6 MIN. FLAG AND CLEAR 6 MIN. COUNT
NO
L7A
1 HOUR YET ?
NO
PROCESS END OF HOUR
YES
L7C
SET 1 HOUR FLAGS
CLEAR MINUTES AND 6 MIN. COUNTERS
INCREMENT HOURS
CHECK FOR DAYLIGHT SAVING TIME ADJUSTMENT
IS IT 2:00 a.m. ?
NO
YES
IS IT SUNDAY ?
NO
YES
10
11
12

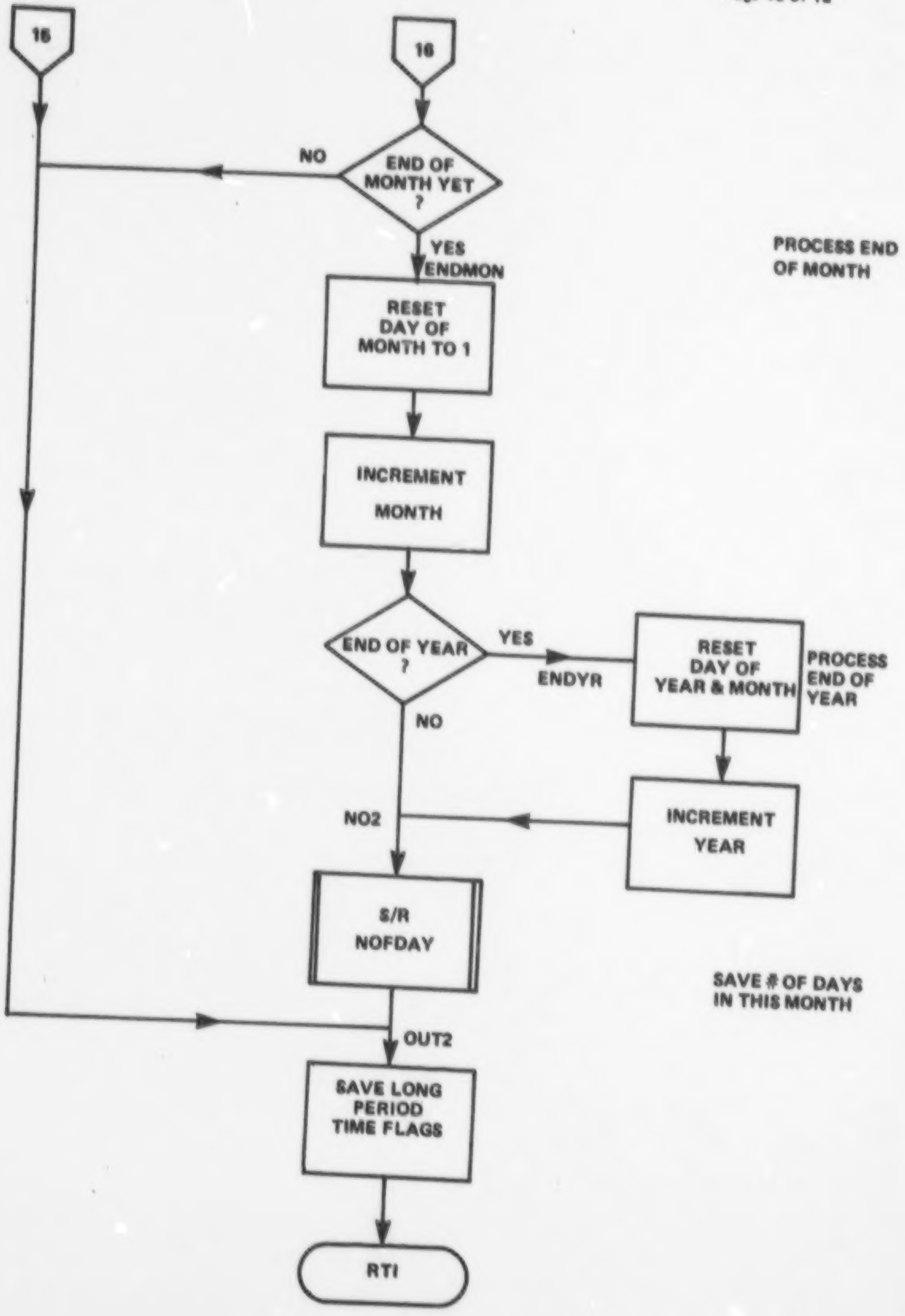
CLOCK ROUTINE (CLOCK)

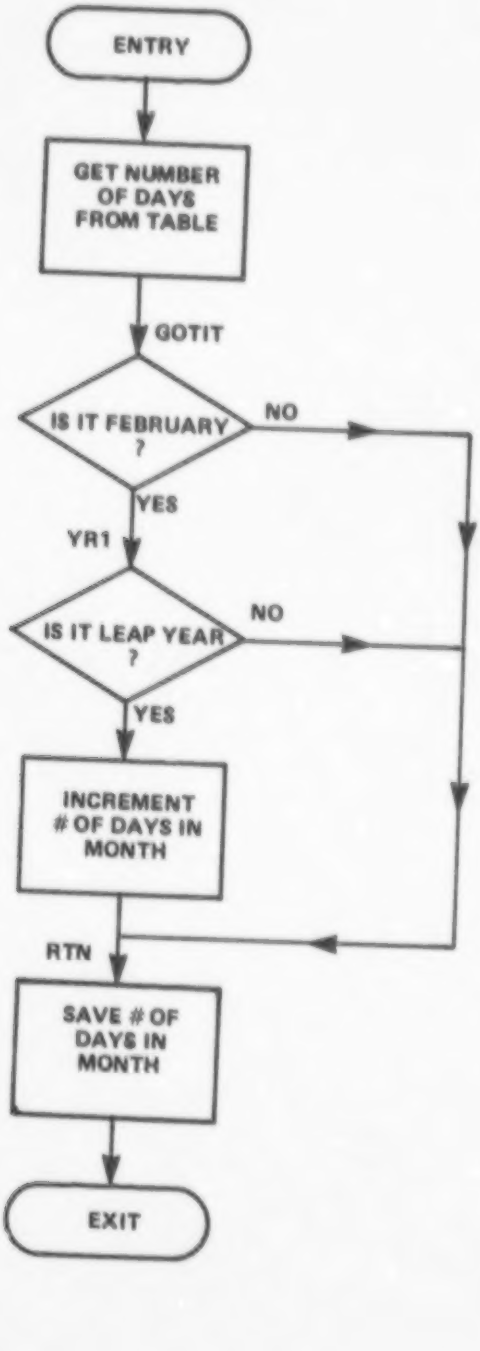

10
11
12
TIME ALREADY CORRECTED ?
YES
NO
S/R DAYLIT
DAYLIGHT SAVING TIME ADJUSTMENT
CLEAR TIME FIX FLAG
L10
END OF DAY ?
NO
YES
L10A
PROCESS END OF DAY
SET END OF DAY FLAGS
SET TIME FIX FLAG (ENABLE TIME CHANGE)
13
14

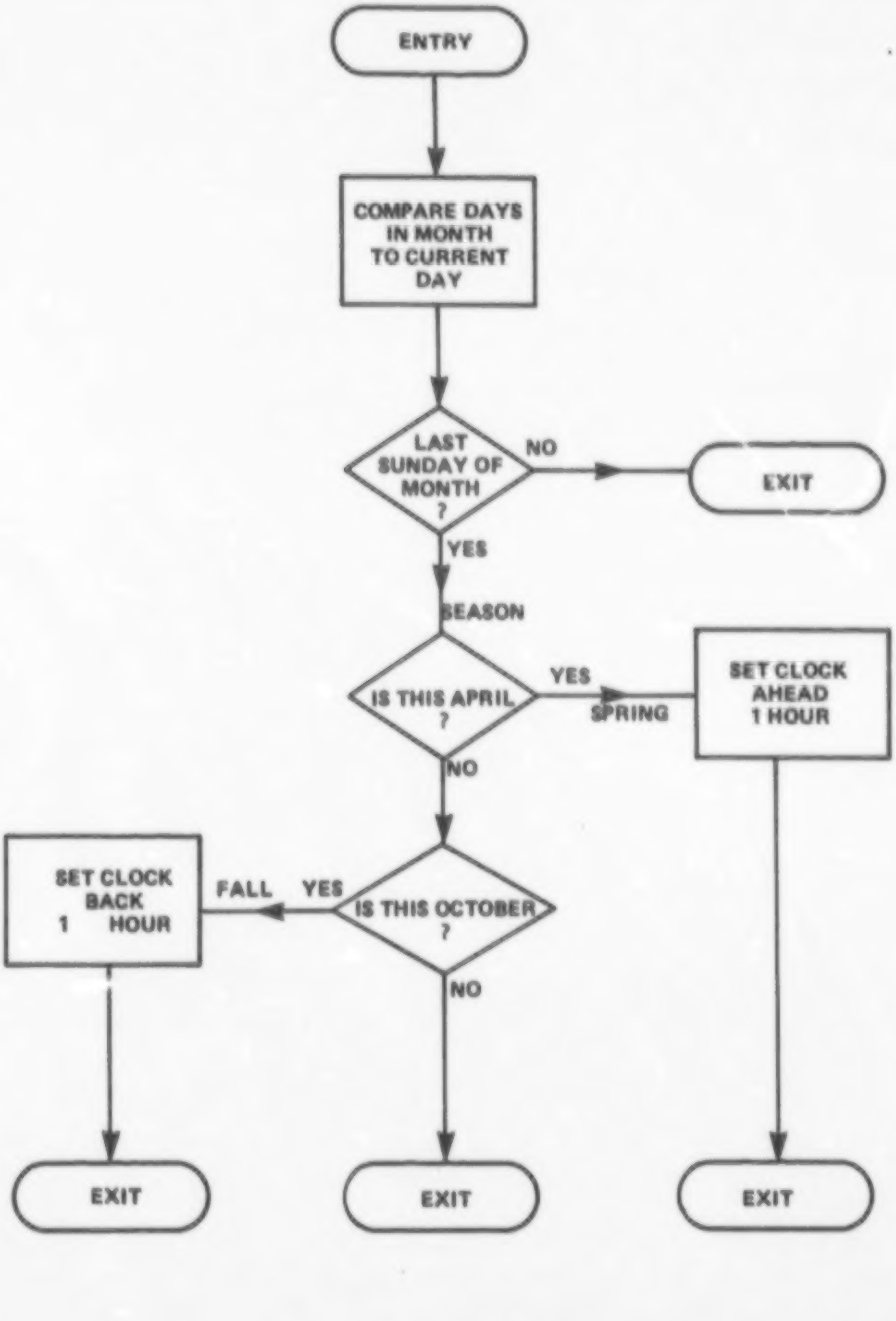
CLOCK ROUTINE (CLOCK)


13
14
CLEAR HOURS
INCREMENT DAY OF WEEK & SET POINTER
INCREMENT DAY OF MONTH & DAY OF YEAR
L30
END OF WEEK ?
NO
YES
PROCESS END OF WEEK
RESET DAY OF WEEK (TO MONDAY)
MON
S/R NOFDAY
GET # OF DAYS IN THIS MONTH
15
16

CLOCK ROUTINE (CLOCK)


15
16
END OF MONTH YET ?
NO
YES
ENDMON
PROCESS END OF MONTH
RESET DAY OF MONTH TO 1
INCREMENT MONTH
END OF YEAR ?
YES
ENDYR
RESET DAY OF YEAR & MONTH
PROCESS END OF YEAR
NO
INCREMENT YEAR
NO2
S/R NOFDAY
SAVE # OF DAYS IN THIS MONTH
OUT2
SAVE LONG PERIOD TIME FLAGS
RTI


ENTRY
GET NUMBER OF DAYS FROM TABLE
GOTIT
IS IT FEBRUARY ?
NO
YES
YR1
IS IT LEAP YEAR ?
NO
YES
INCREMENT # OF DAYS IN MONTH
RTN
SAVE # OF DAYS IN MONTH
EXIT


ENTRY
COMPARE DAYS IN MONTH TO CURRENT DAY
LAST SUNDAY OF MONTH ?
NO
EXIT
YES
SEASON
IS THIS APRIL ?
YES
SPRING
SET CLOCK AHEAD 1 HOUR
NO
IS THIS OCTOBER ?
YES
FALL
SET CLOCK BACK 1 HOUR
NO
EXIT
EXIT
EXIT

7.5/ Keyboard Routine (KEYBD)

Purpose

This routine is executed every 1/30 sec and handles the keyboard input and display functions. It accepts keypushes from the keyboard and takes appropriate action. It then displays the appropriate information on the display. Refer to the keyboard user documentation for details on the uses of the keyboard and display.

Description

The keyboard permits entry of the 16 hexadecimal numbers 0, 1, 2, 3, ..., E, and F. The display consists of a 2-digit upper display, a 4-digit lower display and 10 lights. Each display digit can display any hexadecimal nibble (0-F). The entire upper display and/or the entire lower display can be blanked out. Any of the 10 display lights can be turned on or off. The processing performed by the keyboard routine is based on the "state" concept. At any given time, the keyboard is in a certain state. After a key is pushed (or sometimes after a certain length of time) the keyboard goes to a new state. This new state depends on both the old state and the key pressed. Below is a description of the 10 possible keyboard states and the appearance of the display while in that state.

State 0 -- blank display
  display lights: all off
  upper display: blank
  lower display: blank

State 1 -- time display
  display lights: one of lights 1-7 is on, showing day of week
  upper display: hh   hours
  lower display: mmss   minutes and seconds

State 2 -- expecting 2nd nibble of address
  display lights: lights 1, 2 and 3 flashing, all others off
  upper display: blank
  lower display: 000w   where w is the key last pressed

State 3 -- expecting 3rd nibble of address
display lights: lights 1 and 2 flashing, all others off
upper display: blank
lower display: 00wx where wx are the last two keys pressed

State 4 -- expecting 4th nibble of address
display lights: light 1 is flashing, all other lights off
upper display: blank
lower display: 0wxy where wxy are the last 3 keys pressed

State 5 -- data display (hex)
display lights: all off
upper display: ij where ij is the data at memory address wxyz
lower display: wxyz where wxyz is the memory address

State 6 -- expecting 1st nibble of data
display lights: lights 1 and 2 flashing, all others off
upper display: 00
lower display: wxyz

State 7 -- expecting 2nd nibble of data
display lights: light 1 flashing, all others off
upper display: 0i where i is the key last pressed
lower display: wxyz

State 8 -- data display (hex and binary)
display lights: light 0 on; lights 1-8 binary data display
upper display: ij
lower display: wxyz

State 9 -- display results of PROM and RAM tests
display lights: results of tests
upper display: blank for PROM results, AA for RAM results
lower display: 600d if both RAM and PROM test results are good,
bAdd if there is a RAM or PROM error

Refer to Figure 29 for the state transitions. Each state (indicated by the circled number) is given a rectangle in which the display is described (lights are not shown). Transitions are indicated by arrows joining two states. Each arrow corresponds to pushing a key on the keyboard (or in one case, the passage of a certain amount of time). The arrows are labelled with those keys that cause the transition. (A range of keys is specified with "-". For example, A-D means one of keys A, B, C or D). There are two types of arrows:

- Solid arrows -- transition always occurs when the key(s) labelling the arrow is pushed.
- Dashed arrows -- transition occurs only if key is pressed and the stated condition is satisfied. If the condition is not satisfied, the keyboard reverts to state 1 (time mode).

Also, there is an implied transition from any state to state 0 if 255 minutes pass and no key is pressed (the CLOCK routine takes care of this).

The keyboard routine performs essentially three steps:
1/ process the keypush (if any),
2/ update the display buffers, and
3/ move display buffers to output ports.

The display buffers are, in fact, updated in both Steps 1 and 2. A keypush usually implies a state change which in turn implies a display change. Thus, the display buffers may need some modification in Step 1. These modifications need only be done once, during the transition, such as shifting the lower display when the third address nibble is entered. Step 2, however, performs those display updates that must be done periodically, regardless of whether a key was or was not pressed. For example, the time display changes every 1 second, flashing display lights must be taken care of and data displays must be refreshed since memory might be changing. It is only after all the updates to the display buffers have been made that they are actually moved to the output ports (Step 3). This is when they actually appear to the user. The three steps are described below in a little more detail.

Step 1 -- process the keypush (if any)
The keyboard input port is examined to see if a key is now being held down. If not, the last key press indicator (LSTKEY) is cleared and processing continues at Step 2.

If a key is being pressed, then LSTKEY is examined to determine if a key was being pressed 1/30 sec ago. If there was such a press, then the user has not yet released the key so the keypress was already processed (when the user first pressed it) and processing continues at Step 2. Otherwise, this is a new keypress so LSTKEY is set to 1. Then, the routine to execute when a key is pushed in the current state (indicated by KYSTAT) is extracted from a table and executed. This subroutine will be one of PUSH0, PUSH1, PUSH2, ..., PUSH8, PUSH9. Note that some of these routines do the same thing as others.

Step 2 -- update the display buffers
The routine to execute for the current state display is fetched and executed. It is one of DISPL0, DISPL1, ..., DISPL9. All state displays require some periodic display updating (flashing lights, memory display, time display, etc.).

Step 3 -- move display buffers to output ports
The display buffers are the six variables DISP1, DISP2, ..., DISP6 for the character displays and the two variables LITES and LITES2 for the 10 display lights. The variables DISP1 to DISP6 and LITES2 are output to the display port in conjunction with the display mask DSPMSK (which facilitates blanking of the upper and lower displays where necessary). LITES is then put on the lights display port to turn on or off lights 0 to 7.

The following pages describe the various display and keypush routines for the 10 different keyboard states.

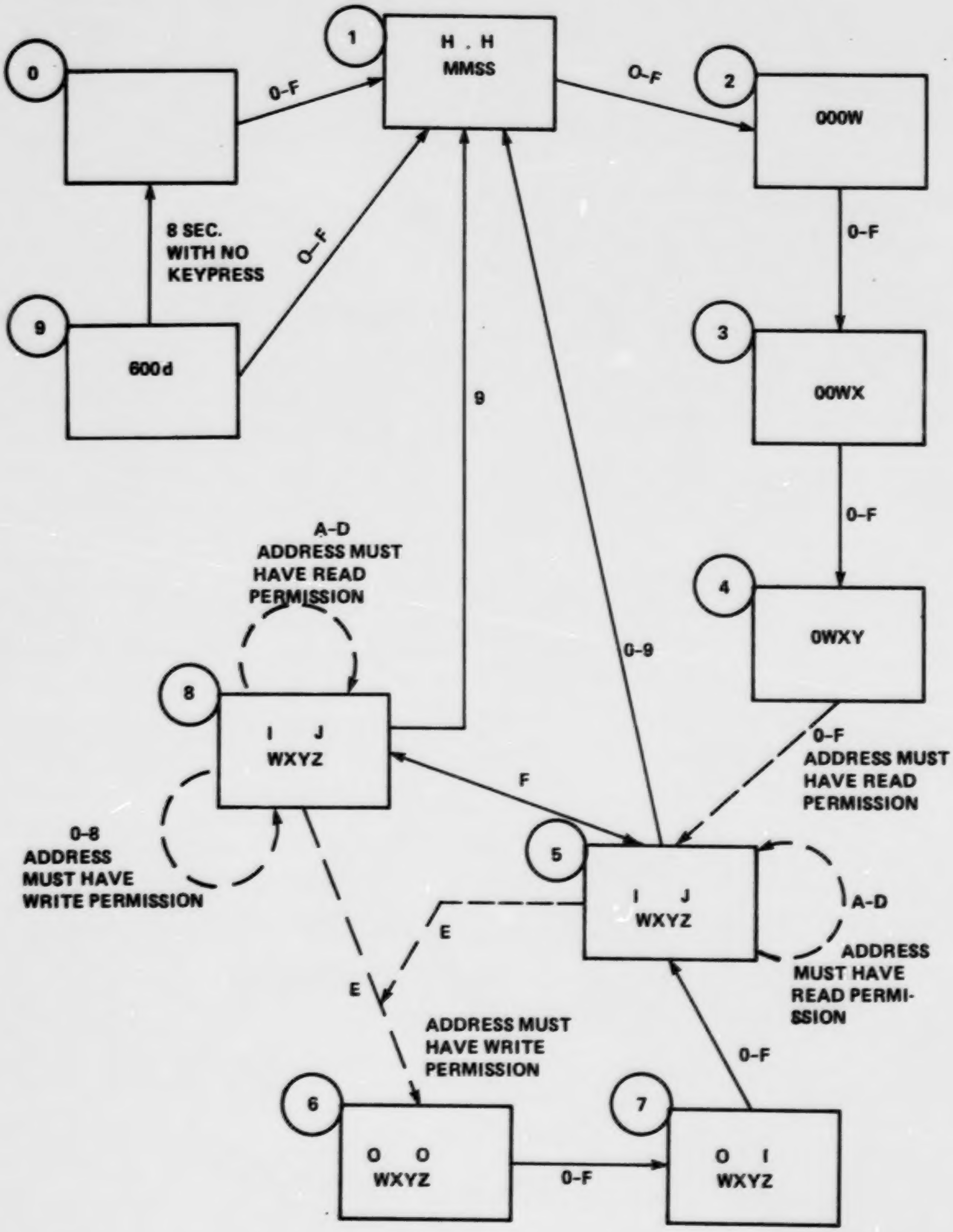

0
1
H . H
MMSS
2
000W
3
00WX
4
0WXY
5
I J
WXYZ
6
O O
WXYZ
7
O I
WXYZ
8
I J
WXYZ
9
600d
0-F
0-F
0-F
0-F
0-F
8 SEC.
WITH NO
KEYPRESS
9
0-9
A-D
ADDRESS MUST
HAVE READ
PERMISSION
0-F
ADDRESS MUST
HAVE READ
PERMISSION
F
0-8
ADDRESS
MUST HAVE
WRITE PERMISSION
A-D
ADDRESS
MUST HAVE
READ PERMI-
SSION
E
E
ADDRESS MUST
HAVE WRITE
PERMISSION
0-F
0-F

Figure 29/ Keyboard Transition Diagram

7.5.1/ Display for State 0 Subroutine (DISPL0)

Description

This subroutine (called by KEYBD) sets the display mask to blank out both the upper and lower displays. It also turns off all 10 display lights.

7.5.2/ Display for State 1 Subroutine (DISPL1)

Description

This subroutine (called by KEYBD) displays the time on the display. The hours are put on the upper display and the minutes and seconds are put on the lower display. Also, the day of week bit pointer DAYPTR is displayed on the display lights.

7.5.3/ Display for State 2 Subroutine (DISPL2)

Description

This subroutine (called by KEYBD) causes lights 1, 2 and 3 to blink periodically. These three flashing lights indicate to the keyboard user that he must still enter three more nibbles of the address.

7.5.4/ Display for States 3 and 6 Subroutine (DISPL3, DISPL6)

Description

This subroutine (called by KEYBD) causes lights 1 and 2 to blink periodically. This is done to indicate to the keyboard user that two more nibbles of data must be entered (address nibbles when state = 3, data nibbles when state = 6).

7.5.5/ Display for States 4 and 7 Subroutine (DISPL4, DISPL7)

Description

This subroutine (called by KEYBD) causes light 1 to blink periodically. This is done to indicate to the keyboard user that one more nibble must be entered (address nibble when state = 4, data nibble when state = 7).

7.5.6/ Display for States 5 and 8 (DISPL5, DISPL8)

Description
This subroutine (called by KEYBD) fetches data from memory (at address ADDRSS) and displays it on the upper display. In addition, if the state is 8, the data byte is displayed on display lights 1-8, each display light being on only if the corresponding bit of the data is set.

7.5.7/ Display for State 9 (DISPL9)

Description
Since state 9 is a temporary state lasting for only about 8 seconds immediately after power-up, this subroutine (called by KEYBD) implements a display timer. If the display timer (DISTMR) has timed out, the blankout state is entered (KYSTAT = 0) and the subroutine terminates.

Otherwise, the display timer is updated. Then, the PROM error flag PRERFL and the RAM error flag RAMERR are examined. If there are no errors at all, the upper display is blanked, the lights are all turned off and 600d is displayed on the lower display (it looks like "good"). The routine then terminates. If there is a memory error, then the results of the tests are displayed. First, "badd" is put on the lower display (not "bad" since the display does not permit the blanking of a single nibble). If there are only RAM errors, "AA" is put on the upper display and the RAM error bit map RAMERR is displayed on lights 0-9.

If there are only PROM errors, the upper display is blanked and the PROM error bit map PRERFL is put on the display lights. If there are both RAM and PROM errors, then both these actions are taken (the PROM results for the first 4 seconds, the RAM results for the next 4 seconds). This, of course, utilizes the display timer.

7.5.8/ Key Pushed While in States 0 or 9 Subroutine (PUSH0, PUSH9)

Description
This subroutine (called by KEYBD) simply causes the keyboard to enter time mode (KYSTAT = 1). It then enables the upper and lower displays,

which were previously blanked out (for state 0) or partially blanked out (for state 9).

7.5.9/ Key Pushed While in State 1 Subroutine (PUSH1)

Description
The key just pushed is the first nibble of the address. The upper display is blanked out and the lower display is set to zeros. Then, the key is displayed in the bottom right corner of the display. The next keyboard state (KYSTAT = 2) is then entered.

7.5.10/ Key Pushed While in States 2 or 3 Subroutine (PUSH2, PUSH3)

Description
This subroutine is called by KEYBD. Another nibble of the address has just been entered. The lower display is shifted one nibble to the left and the new key just pressed is displayed in the bottom right corner. The state is then incremented (2 goes to 3, 3 goes to 4).

7.5.11/ Key Pushed While in State 4 Subroutine (PUSH4)

Description
This subroutine is called by KEYBD. The last nibble of the address has just been entered. The lower display is shifted one nibble to the left and the key pressed is displayed at the bottom right corner. The complete address is now on the lower display (or at least, it will be once the display buffers are output). The next state is then entered (KYSTAT = 5). The 4 nibbles are merged into a 2-byte address ADDRSS. The lights are turned off and the upper display (formerly blank) is enabled. Then, subroutine READOK is invoked to determine if the address has read permission. If it does not, time mode is entered (KYSTAT = 0). If the address has read permission, the data are fetched from that address and displayed on the upper display.

7.5.12/ Key Pushed While in State 5 Subroutine (PUSH5)

Description
This subroutine is called by KEYBD. The processing outlined below is

performed, depending on which key was pressed.

Key Pressed is Between 0 and 9 (inclusive) -- The keyboard reverts to time mode (state 1).

The Key Pressed is E -- Subroutine WRITOK is invoked to determine if this address may be changed. If write permission is denied, time mode is entered and the subroutine terminates. Otherwise, the upper display is set to zeros and the next state (KYSTAT = 6) is entered.

The Key Pressed is F -- Display light 0 is turned on to indicate hex/binary display mode. The hex/binary display state (8) is then entered.

The Key Pressed is Between A and D (inclusive) -- Subroutine ADDCHG is invoked to change the address and the routine terminates.

7.5.13/ Key Pushed While in State 6 Subroutine (PUSH6)

Description

This subroutine is called by KEYBD. The key pressed is displayed on the upper right display and the next state (KYSTAT = 7) is entered.

7.5.14/ Key Pushed While in State 7 Subroutine (PUSH7)

Description

This subroutine is called by KEYBD. The key just pushed is combined with the key last pushed to form a data byte. This byte is then written to the memory address indicated by ADDRSS. The display lights are turned off (they were blinking to request the last nibble) and the keyboard returns to state 5.

7.5.15/ Key Pushed While in State 8 Subroutine (PUSH8)

Description

This subroutine is called by KEYBD. The processing done depends on which key was pressed, as outlined below.

Key Pressed Less Than 9 -- This is an attempt to change memory. First, subroutine WRITOK is invoked to make sure the change is allowed. If it is not, the keyboard reverts to time mode. Otherwise, the change is allowed. If the key pressed was 0, then the memory address (ADDRSS) is cleared. If the key was 1-8, then the appropriate bit of memory is toggled.

Key Pressed is 9 -- The display reverts to time mode.

Key Pressed is E -- This also is an attempt to change memory. Subroutine WRITOK is invoked to determine if the change is allowed. If it is not permitted, the keyboard reverts to time mode and the subroutine terminates. Otherwise, the display lights are all turned off, the upper display is filled with zeros and the next state (KYSTAT = 6) is entered.

Key Pressed is F -- This causes the keyboard routine to leave hex/binary display mode. The display lights are all turned off and state 5 is entered.

Key Pressed is Between A and D (inclusive) -- Subroutine ADDCHG is invoked to processs the address change.

7.5.16/ Process Address Change Subroutine (ADDCHG)

Purpose

This subroutine is invoked by KEYBD whenever the user pushes one of keys A, B, C or D while in state 5 or state 8. These keys are used to either increment or decrement the current address displayed.

Description

Depending on the key pushed, the following actions are taken (performed on ADDRSS):

- A key pressed -- increment address by 1
- B key pressed -- decrement address by 1
- C key pressed -- increment address by $10
- D key pressed -- decrement address by $10

Subroutine READOK is then invoked to determine if the new address may be read. If not, time mode is entered (KYSTAT = 1) and the subroutine terminates. Otherwise, the read is allowed so the new address is put on the lower display and the routine terminates.

7.5.17/ Determine If Address Can Be Read Subroutine (READOK)

Purpose

This subroutine (called by KEYBD) determines if the address contained in ADDRSS can be displayed at the keyboard.

Description

The following addresses can be read:

- any RAM address
- any PROM address ($C000-$FFFF)
- detector ports ($5001-$5006)

Any other address will cause read permission to be denied.

7.5.18/ Determine If Memory Can Be Changed Subroutine (WRITOK)

Purpose

This subroutine is called by KEYBD and it determines if the address contained in ADDRSS is allowed to be changed. Some memory changes could "hang" the controller or destroy important RAM variables, so it is in the best interest of a smoothly operating system that the critical RAM locations be protected from the unknowing keyboard user.

Description

First, the two byte security code SECCOD can always be changed. If ADDRSS contains the address of SECCOD, a change is permitted and the routine terminates. This also occurs if ADDRSS contains the address of SECCOD+1.

Otherwise, the security code itself is examined. If the value is not correct, the user has not yet entered the security code so no change is permitted.

If the address is not within RAM, the change is prohibited and the routine terminates.

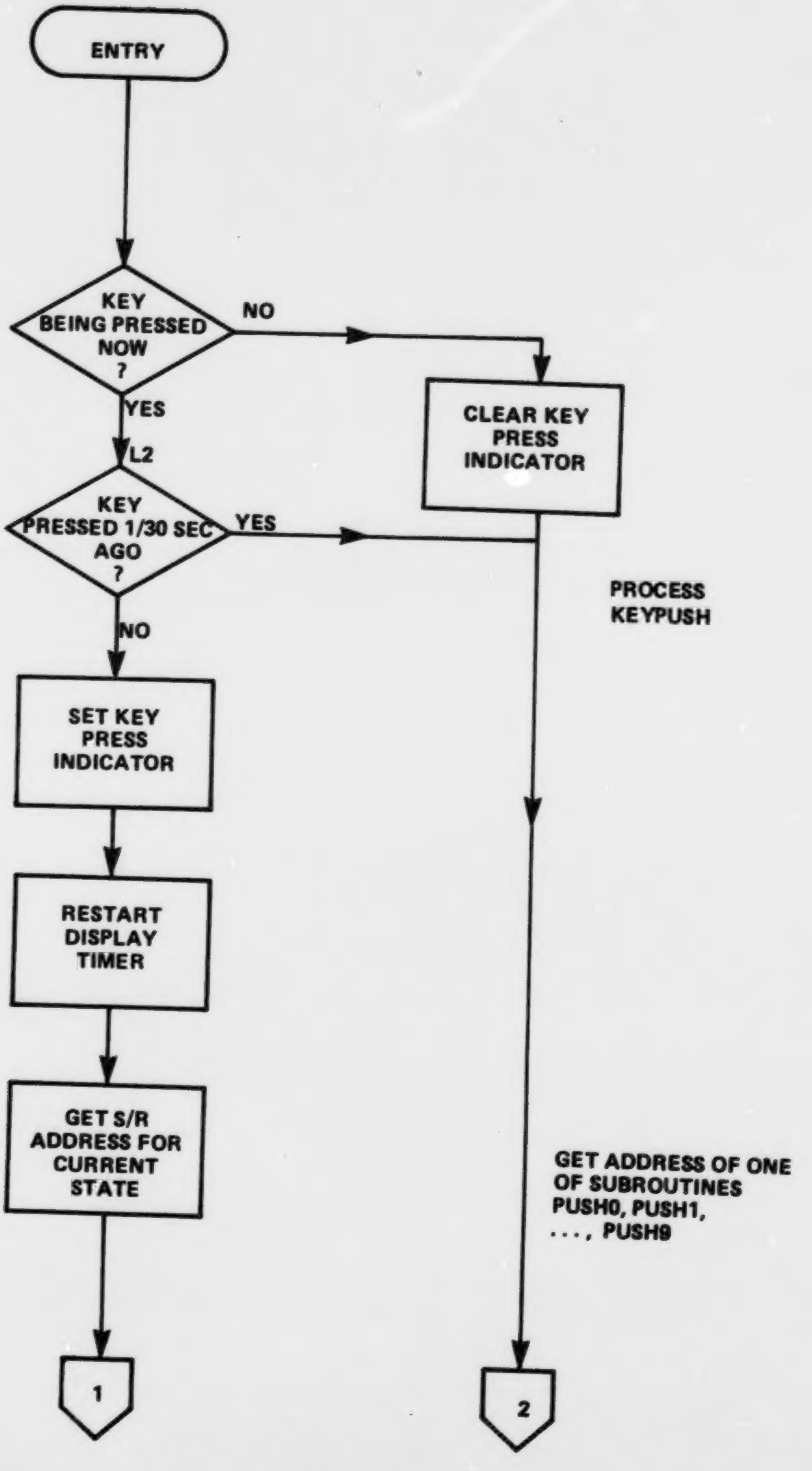
If the address is within RAM but also within the system stack, the change is prohibited and the routine terminates.

In addition, the address is checked against a list of special, protected RAM locations. If the address is found in the list, the change is not allowed and the routine terminates. Otherwise, the change is permitted.

(Refer to the program listing for a list of the specifically protected locations.)

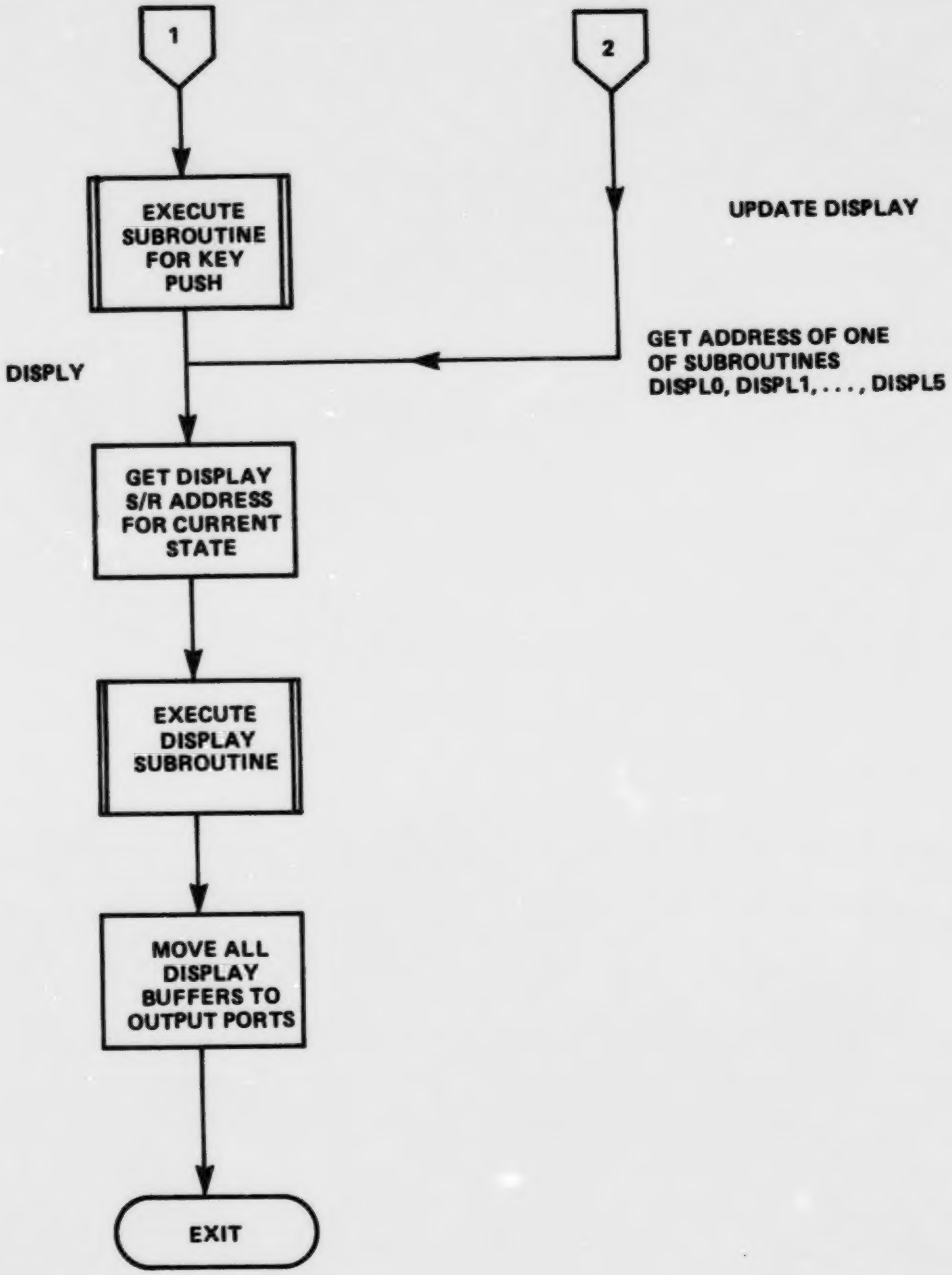
KEYBOARD ROUTINE (KEYBD)

Page 1 of 24

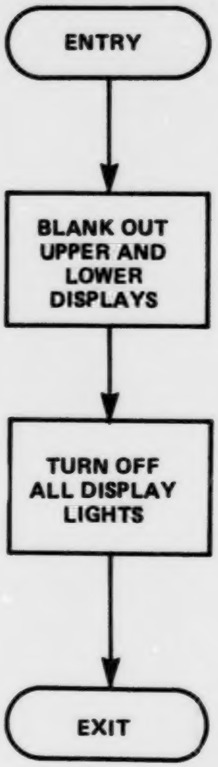

ENTRY
KEY BEING PRESSED NOW ?
NO
YES
L2
CLEAR KEY PRESS INDICATOR
KEY PRESSED 1/30 SEC AGO ?
YES
NO
PROCESS KEYPUSH
SET KEY PRESS INDICATOR
RESTART DISPLAY TIMER
GET S/R ADDRESS FOR CURRENT STATE
GET ADDRESS OF ONE OF SUBROUTINES PUSH0, PUSH1, . . . , PUSH9
1
2

KEYBOARD ROUTINE (KEYBD)

Page 2 of 24


1
2
EXECUTE SUBROUTINE FOR KEY PUSH
UPDATE DISPLAY
GET ADDRESS OF ONE OF SUBROUTINES DISPL0, DISPL1, . . . , DISPL5
DISPLY
GET DISPLAY S/R ADDRESS FOR CURRENT STATE
EXECUTE DISPLAY SUBROUTINE
MOVE ALL DISPLAY BUFFERS TO OUTPUT PORTS
EXIT

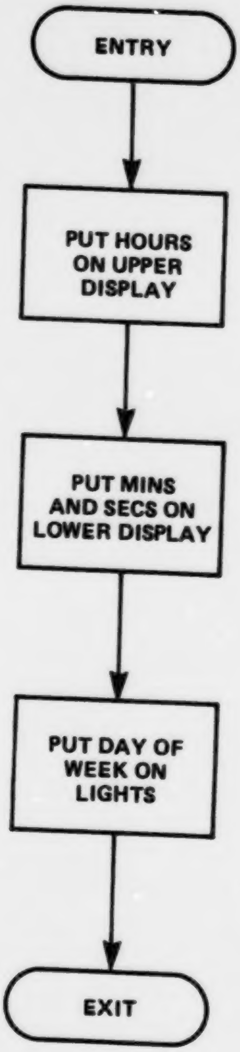
KEYBOARD ROUTINE (KEYBD)
DISPLAY FOR STATE 0 SUBROUTINE (DISPL0)


ENTRY
BLANK OUT UPPER AND LOWER DISPLAYS
TURN OFF ALL DISPLAY LIGHTS
EXIT

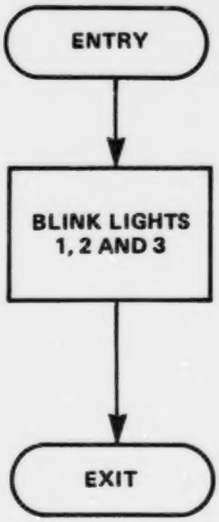
BLANK ENTIRE DISPLAY

KEYBOARD ROUTINE (KEYBD)

DISPLAY FOR STATE 1 SUBROUTINE (DISPL1)

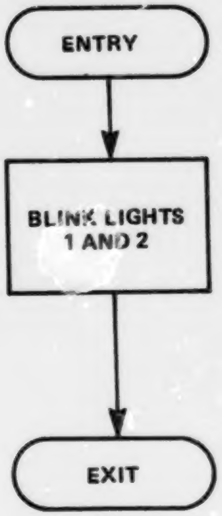

ENTRY
PUT HOURS ON UPPER DISPLAY
PUT MINS AND SECS ON LOWER DISPLAY
PUT DAY OF WEEK ON LIGHTS
EXIT

DISPLAY TIME


ENTRY
BLINK LIGHTS 1, 2 AND 3
EXIT

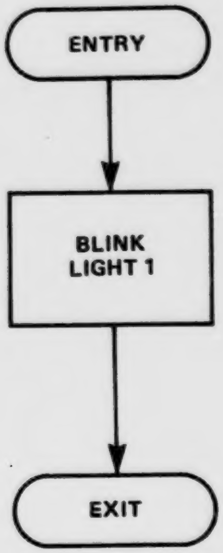
INDICATE 3 MORE NIBBLES EXPECTED

KEYBOARD ROUTINE (KEYBD)
DISPLAY FOR STATE 3 and 6 SUBROUTINE (DISPL3, DISPL6)


ENTRY
BLINK LIGHTS 1 AND 2
EXIT

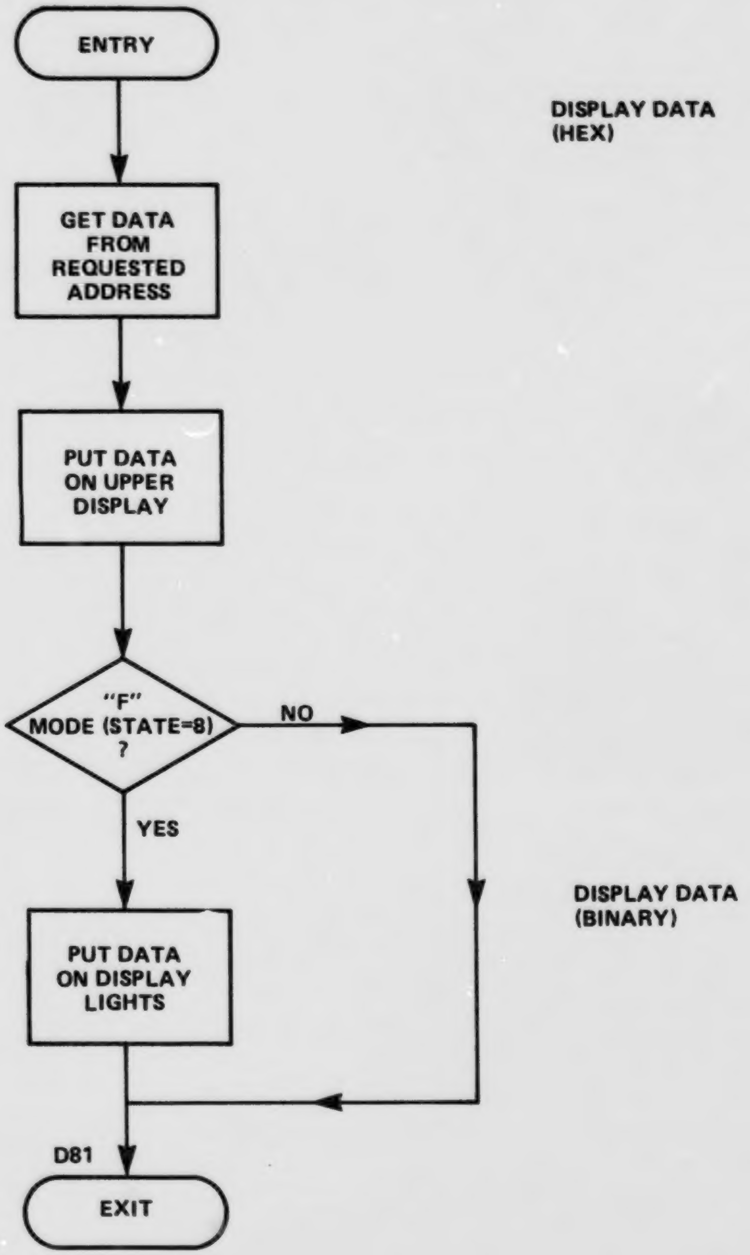
INDICATE 2 MORE NIBBLES EXPECTED

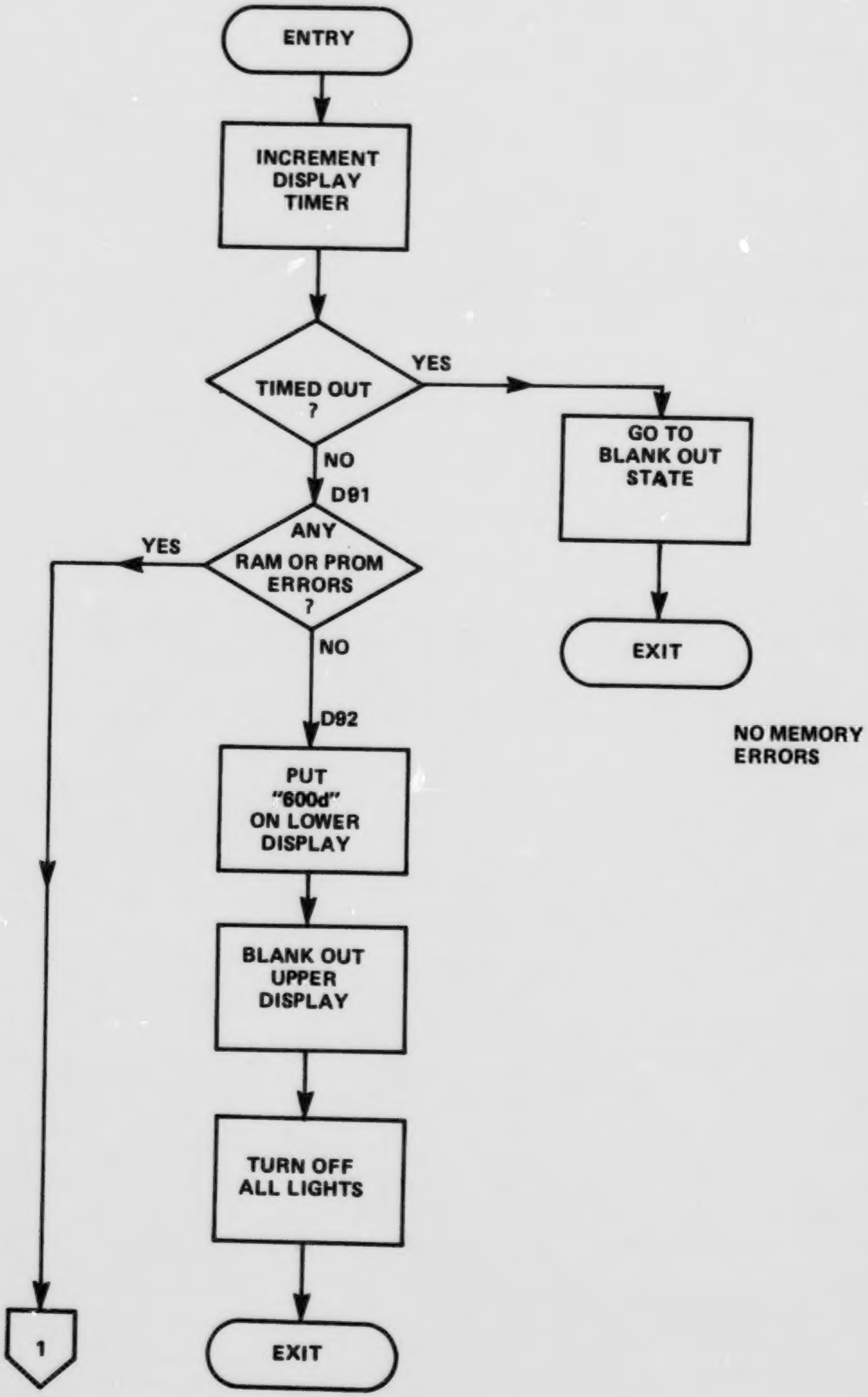
KEYBOARD ROUTINE (KEYBD)
DISPLAY FOR STATE 4 and 7 SUBROUTINE (DISPL4, DISPL7)


ENTRY
BLINK LIGHT 1
EXIT

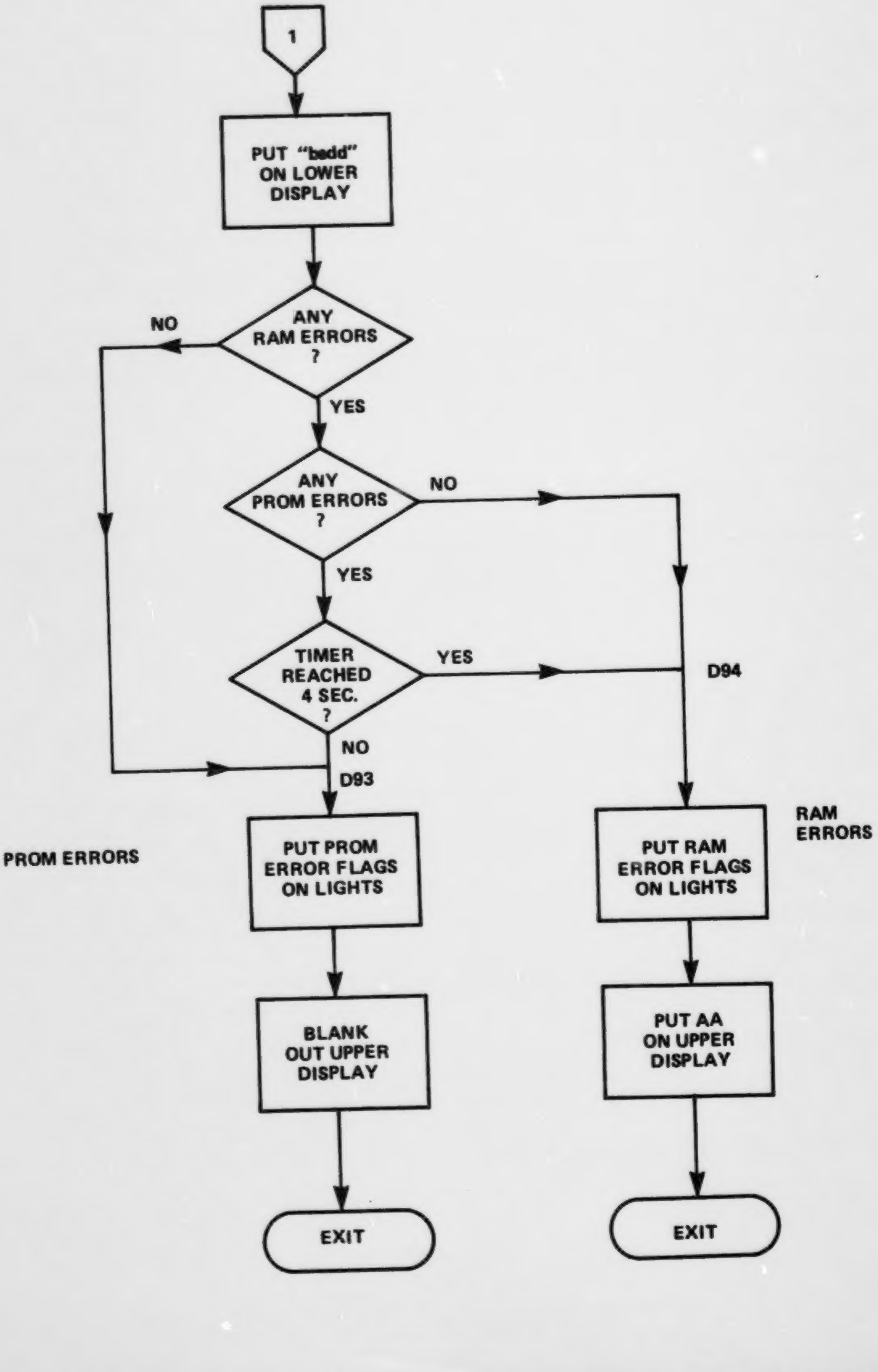
INDICATE 1 MORE NIBBLE EXPECTED

KEYBOARD ROUTINE (KEYBD)
DISPLAY FOR STATES 5 and 8 SUBROUTINE (DISPL5, DISPL8)

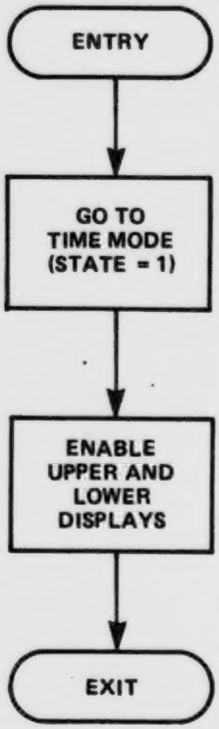

ENTRY
DISPLAY DATA (HEX)
GET DATA FROM REQUESTED ADDRESS
PUT DATA ON UPPER DISPLAY
"F" MODE (STATE=8) ?
NO
YES
DISPLAY DATA (BINARY)
PUT DATA ON DISPLAY LIGHTS
D81
EXIT


ENTRY
INCREMENT DISPLAY TIMER
TIMED OUT ?
YES
GO TO BLANK OUT STATE
EXIT
NO
D91
ANY RAM OR PROM ERRORS ?
YES
NO
D92
NO MEMORY ERRORS
PUT "600d" ON LOWER DISPLAY
BLANK OUT UPPER DISPLAY
TURN OFF ALL LIGHTS
1
EXIT

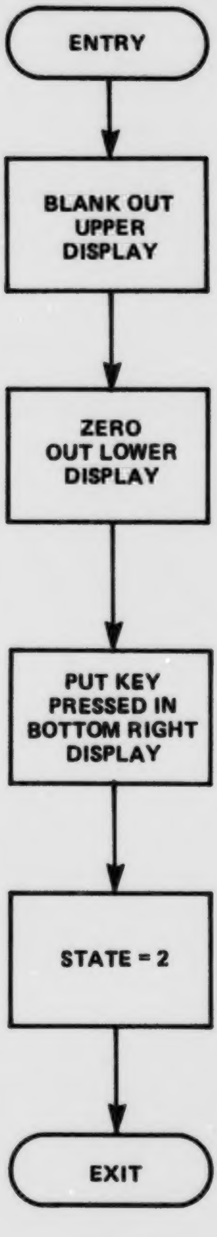
KEYBOARD ROUTINE (KEYBD)
DISPLAY FOR STATE 9 SUBROUTINE (DISPL 9)


1
PUT "bedd" ON LOWER DISPLAY
ANY RAM ERRORS ?
NO
YES
ANY PROM ERRORS ?
NO
YES
TIMER REACHED 4 SEC. ?
YES
D94
NO
D93
PROM ERRORS
PUT PROM ERROR FLAGS ON LIGHTS
RAM ERRORS
PUT RAM ERROR FLAGS ON LIGHTS
BLANK OUT UPPER DISPLAY
PUT AA ON UPPER DISPLAY
EXIT
EXIT

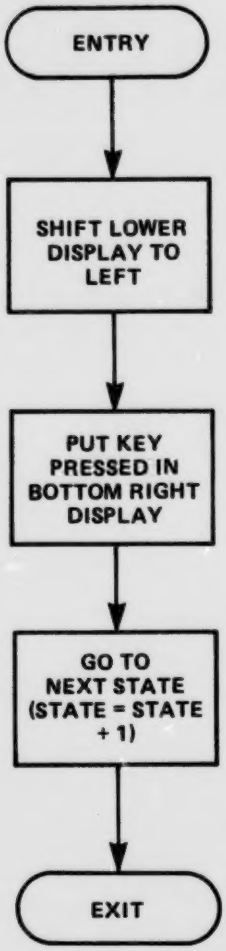
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 0 or 9 SUBROUTINE (PUSH0, PUSH9)


ENTRY
GO TO TIME MODE (STATE = 1)
ENABLE UPPER AND LOWER DISPLAYS
EXIT

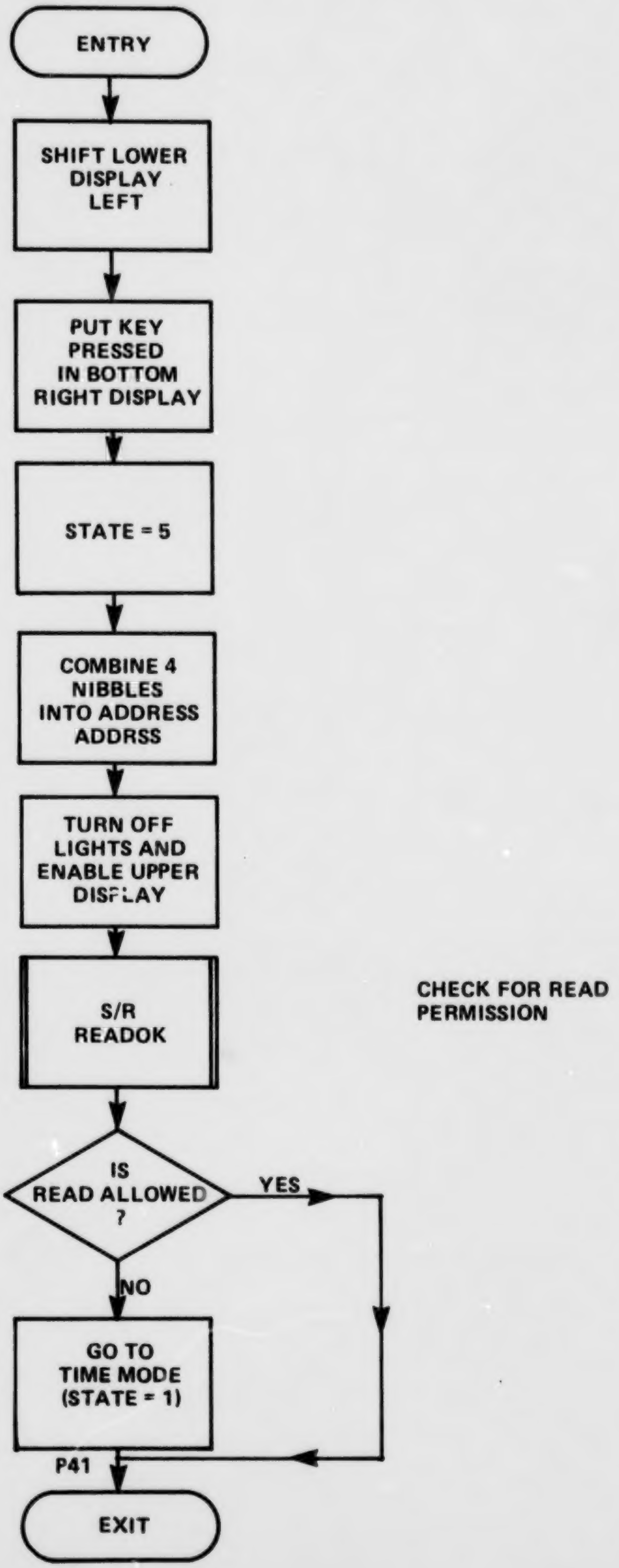
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 1 SUBROUTINE (PUSH1)


ENTRY
BLANK OUT UPPER DISPLAY
ZERO OUT LOWER DISPLAY
PUT KEY PRESSED IN BOTTOM RIGHT DISPLAY
STATE = 2
EXIT

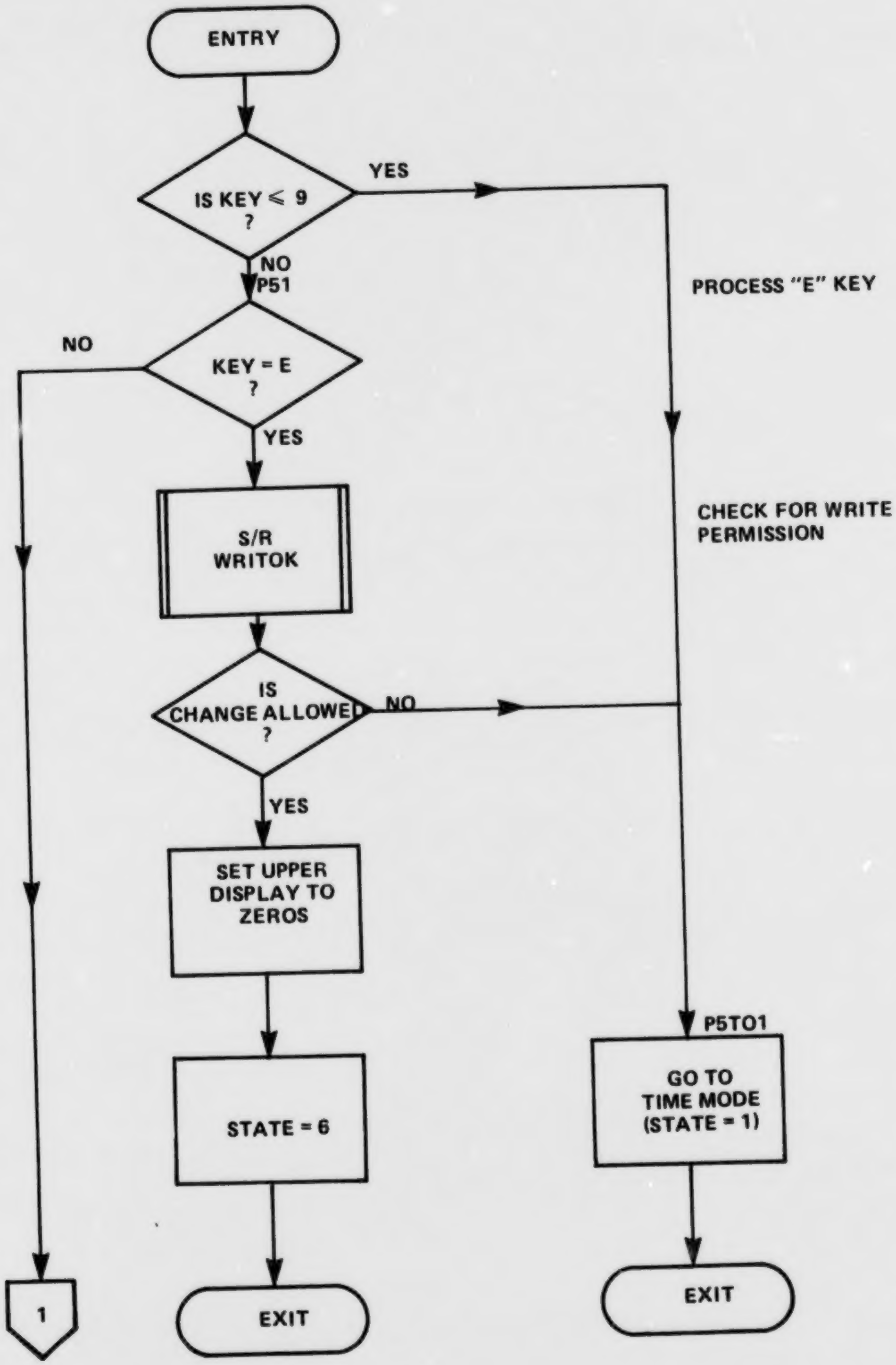
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 2 OR 3 SUBROUTINE (PUSH2, PUSH3)


ENTRY
SHIFT LOWER DISPLAY TO LEFT
PUT KEY PRESSED IN BOTTOM RIGHT DISPLAY
GO TO NEXT STATE (STATE = STATE + 1)
EXIT

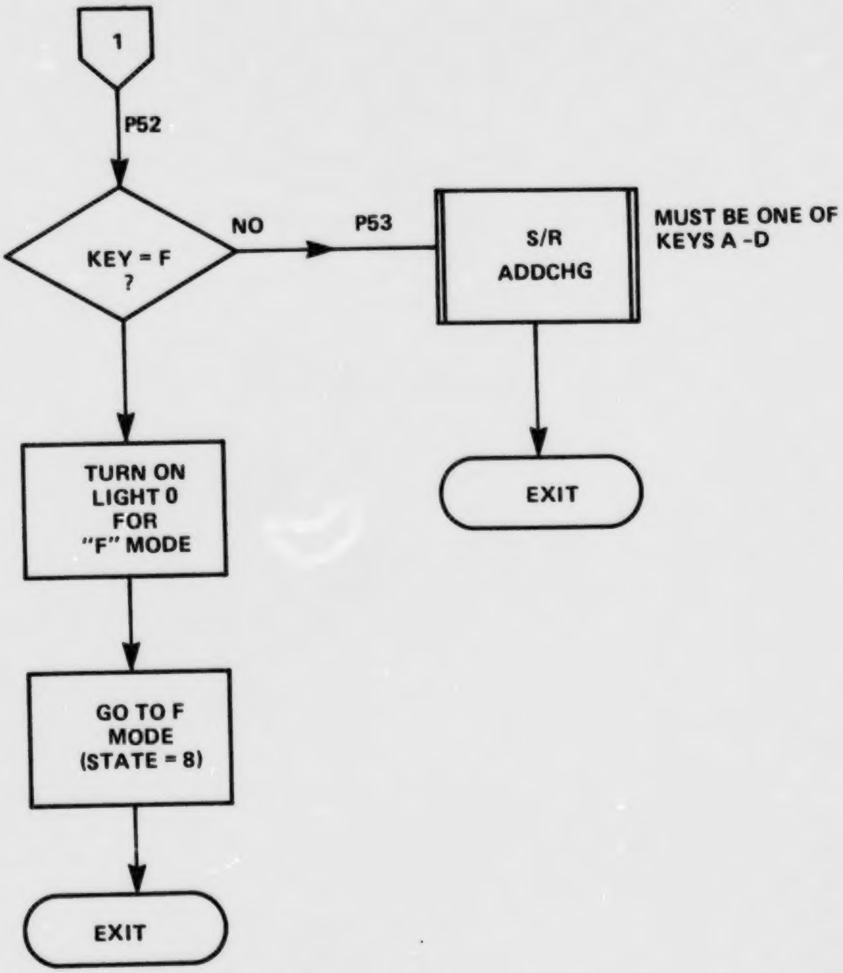
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 4 SUBROUTINE (PUSH4)


ENTRY
SHIFT LOWER DISPLAY LEFT
PUT KEY PRESSED IN BOTTOM RIGHT DISPLAY
STATE = 5
COMBINE 4 NIBBLES INTO ADDRESS ADDRSS
TURN OFF LIGHTS AND ENABLE UPPER DISPLAY
S/R READOK
CHECK FOR READ PERMISSION
IS READ ALLOWED ?
YES
NO
GO TO TIME MODE (STATE = 1)
P41
EXIT

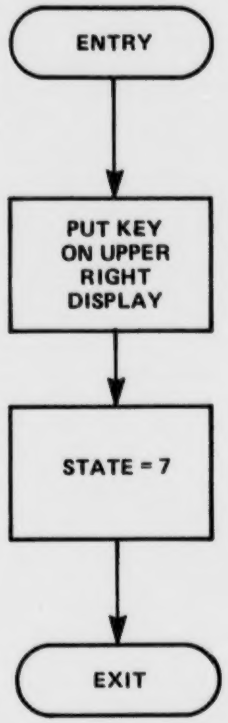
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 5 SUBROUTINE (PUSH5)


ENTRY
IS KEY ≤ 9 ?
YES
NO
P51
PROCESS "E" KEY
KEY = E ?
NO
YES
S/R WRITOK
CHECK FOR WRITE PERMISSION
IS CHANGE ALLOWED ?
NO
YES
SET UPPER DISPLAY TO ZEROS
STATE = 6
P5TO1
GO TO TIME MODE (STATE = 1)
1
EXIT
EXIT

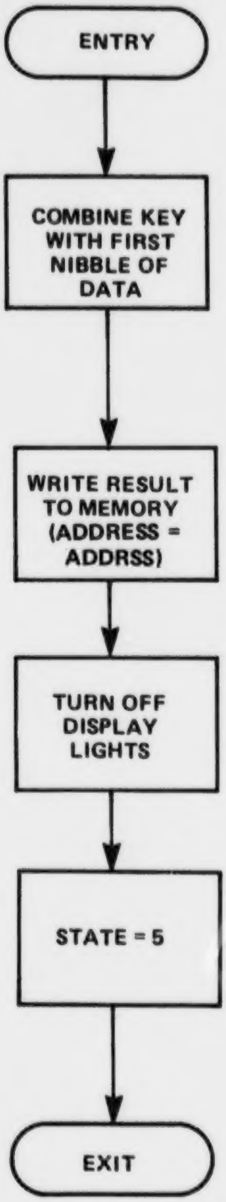
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 5 SUBROUTINE (PUSH 5)


1
P52
KEY = F ?
NO
P53
S/R ADDCHG
MUST BE ONE OF KEYS A -D
EXIT
TURN ON LIGHT 0 FOR "F" MODE
GO TO F MODE (STATE = 8)
EXIT

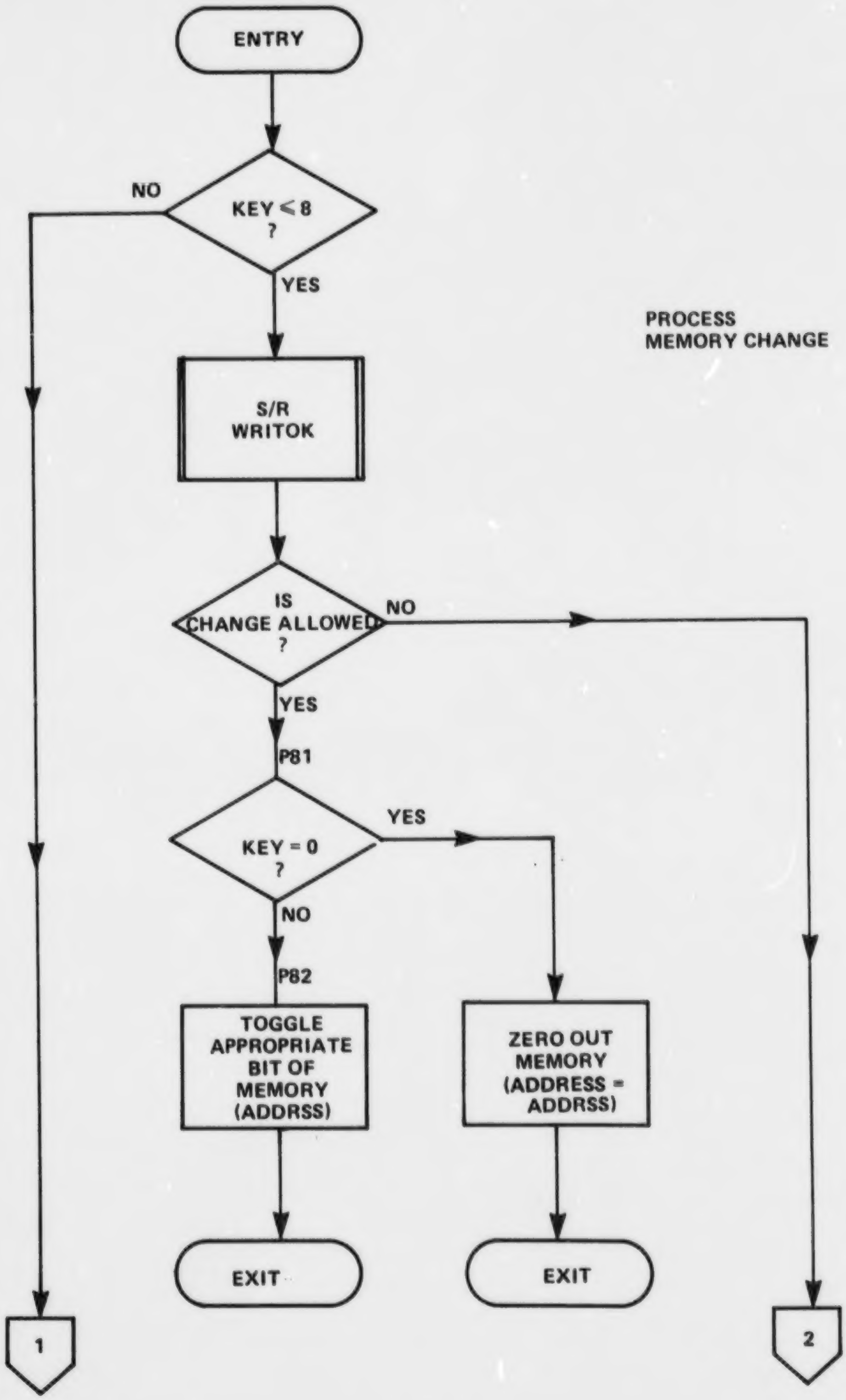
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 6 SUBROUTINE (PUSH6)


ENTRY
PUT KEY ON UPPER RIGHT DISPLAY
STATE = 7
EXIT

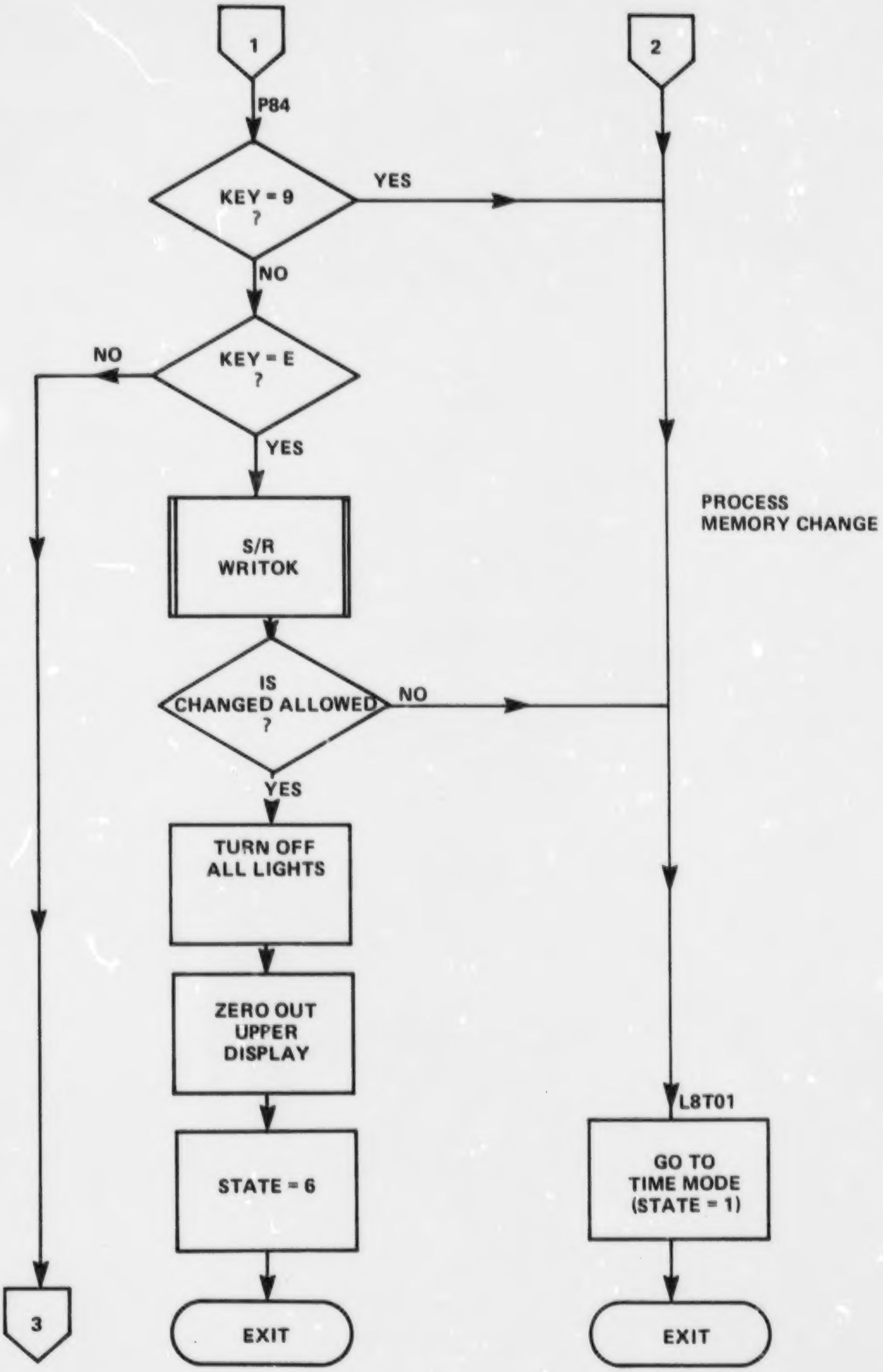
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 7 SUBROUTINE (PUSH7)


ENTRY
COMBINE KEY WITH FIRST NIBBLE OF DATA
WRITE RESULT TO MEMORY (ADDRESS = ADDRSS)
TURN OFF DISPLAY LIGHTS
STATE = 5
EXIT

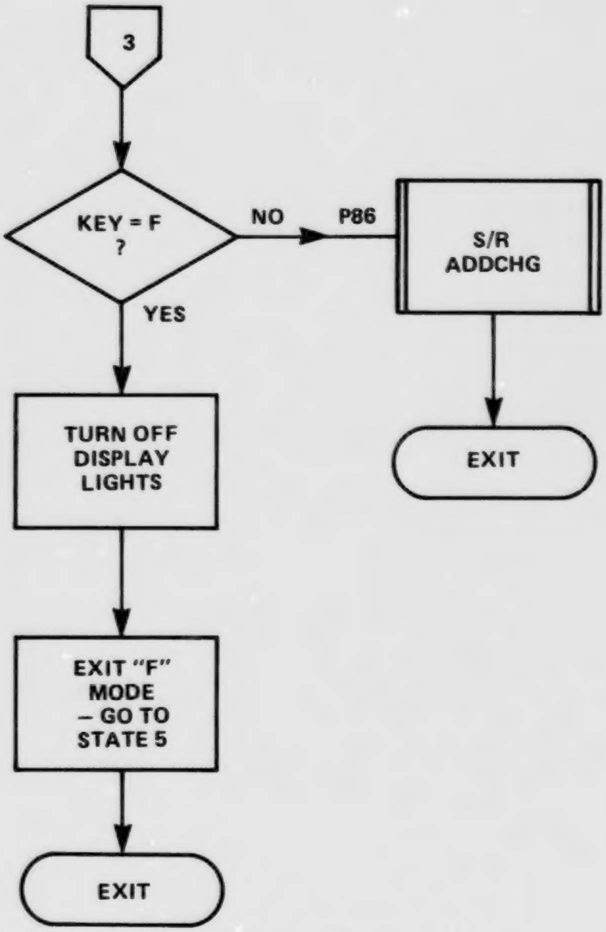
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 8 SUBROUTINE (PUSH8)


ENTRY
KEY ≤ 8 ?
NO
YES
PROCESS MEMORY CHANGE
S/R WRITOK
IS CHANGE ALLOWED ?
NO
YES
P81
KEY = 0 ?
YES
NO
P82
TOGGLE APPROPRIATE BIT OF MEMORY (ADDRSS)
ZERO OUT MEMORY (ADDRESS = ADDRSS)
EXIT
EXIT
1
2

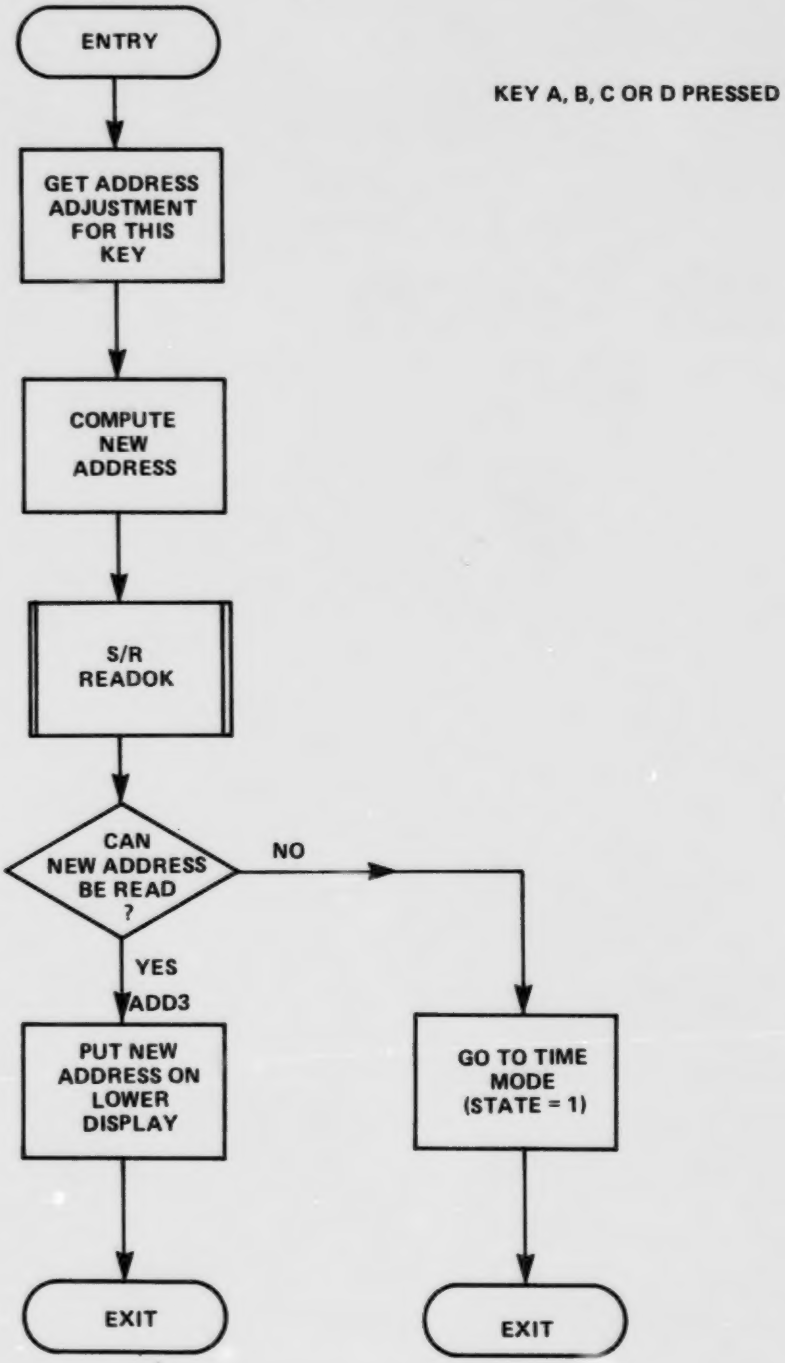
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 8 SUBROUTINE (PUSH 8)


1
P84
KEY = 9 ?
YES
NO
KEY = E ?
NO
YES
S/R WRITOK
IS CHANGED ALLOWED ?
NO
YES
TURN OFF ALL LIGHTS
ZERO OUT UPPER DISPLAY
STATE = 6
EXIT
3
2
PROCESS MEMORY CHANGE
L8T01
GO TO TIME MODE (STATE = 1)
EXIT

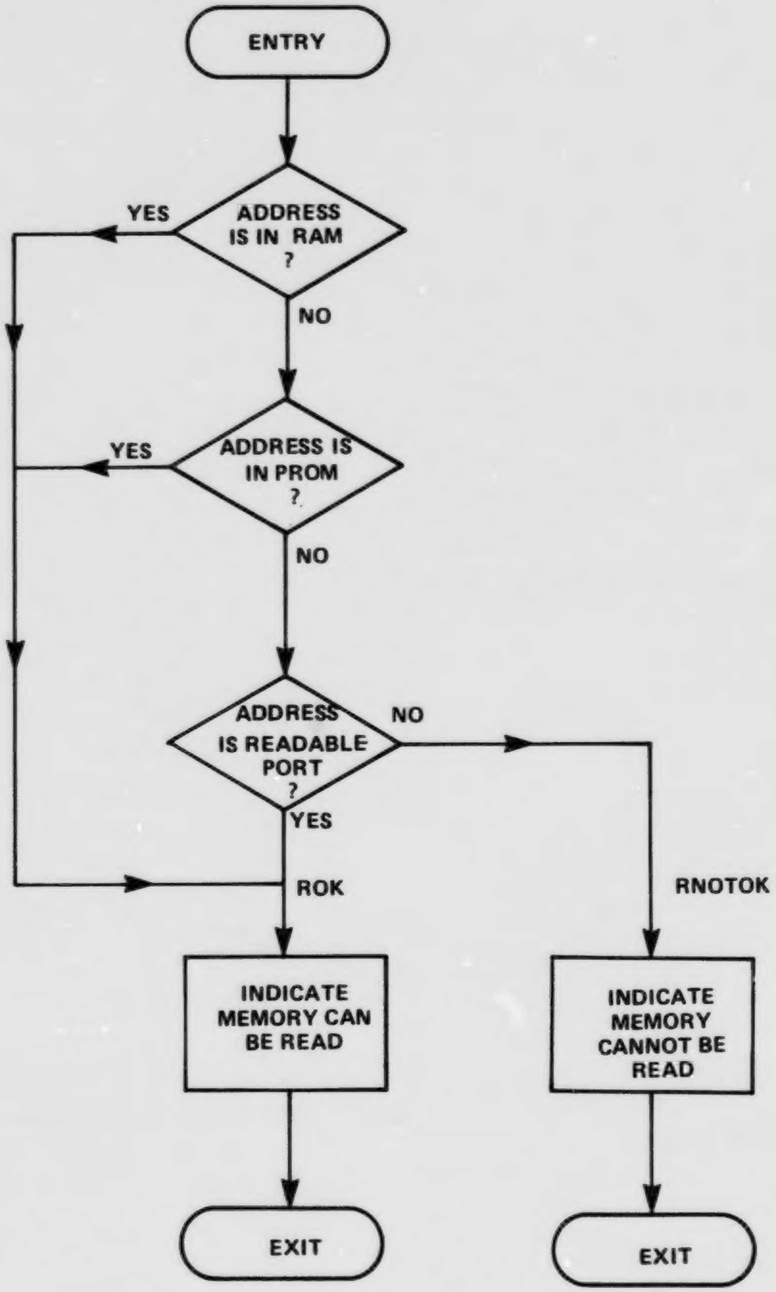
KEYBOARD ROUTINE (KEYBD)
KEY PUSHED WHILE IN STATE 8 SUBROUTINE (PUSH 8)


3
KEY = F ?
NO
P86
S/R ADDCHG
YES
EXIT
TURN OFF DISPLAY LIGHTS
EXIT "F" MODE – GO TO STATE 5
EXIT

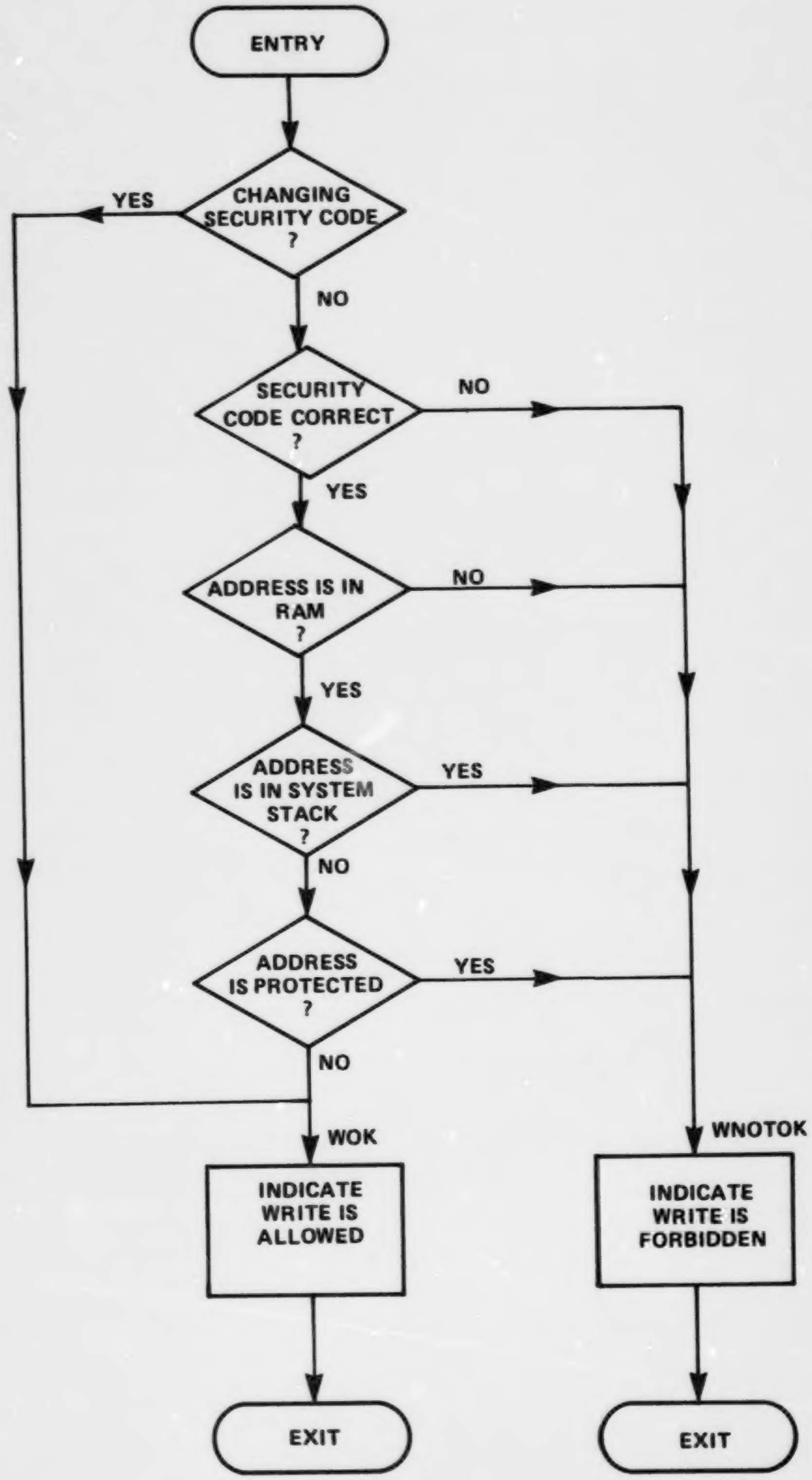
KEYBOARD ROUTINE (KEYBD)
PROCESS ADDRESS CHANGE SUBROUTINE (ADDCHG)


ENTRY
KEY A, B, C OR D PRESSED
GET ADDRESS ADJUSTMENT FOR THIS KEY
COMPUTE NEW ADDRESS
S/R READOK
CAN NEW ADDRESS BE READ ?
NO
YES
ADD3
PUT NEW ADDRESS ON LOWER DISPLAY
GO TO TIME MODE (STATE = 1)
EXIT
EXIT

KEYBOARD ROUTINE (KEYBD)
DETERMINE IF MEMORY CAN BE READ SUBROUTINE (READOK)


ENTRY
ADDRESS IS IN RAM ?
YES
NO
ADDRESS IS IN PROM ?
YES
NO
ADDRESS IS READABLE PORT ?
NO
YES
ROK
RNOTOK
INDICATE MEMORY CAN BE READ
INDICATE MEMORY CANNOT BE READ
EXIT
EXIT

KEYBOARD ROUTINE (KEYBD)
DETERMINE IF MEMORY CAN BE CHANGED SUBROUTINE (WRITOK)


ENTRY
CHANGING SECURITY CODE ?
YES
NO
SECURITY CODE CORRECT ?
NO
YES
ADDRESS IS IN RAM ?
NO
YES
ADDRESS IS IN SYSTEM STACK ?
YES
NO
ADDRESS IS PROTECTED ?
YES
NO
WOK
WNOTOK
INDICATE WRITE IS ALLOWED
INDICATE WRITE IS FORBIDDEN
EXIT
EXIT

7.6/ Communication Routines

7.6.1/ Main Routine (COMM)

Purpose

This routine handles communications between the controller and the central computer. It is called by ITEST whenever an interrupt is received from the communication interface (ACIA).

By examining the ACIA status register, COMM decides which one of the following actions to take:

- invoke transmit routine (XMIT),
- invoke receive routine (RECV),
- reset communications if carrier lost, or
- do nothing as a safety precaution (cause of interrupt unknown)

Refer Appendix A for the format of communication messages.

Description

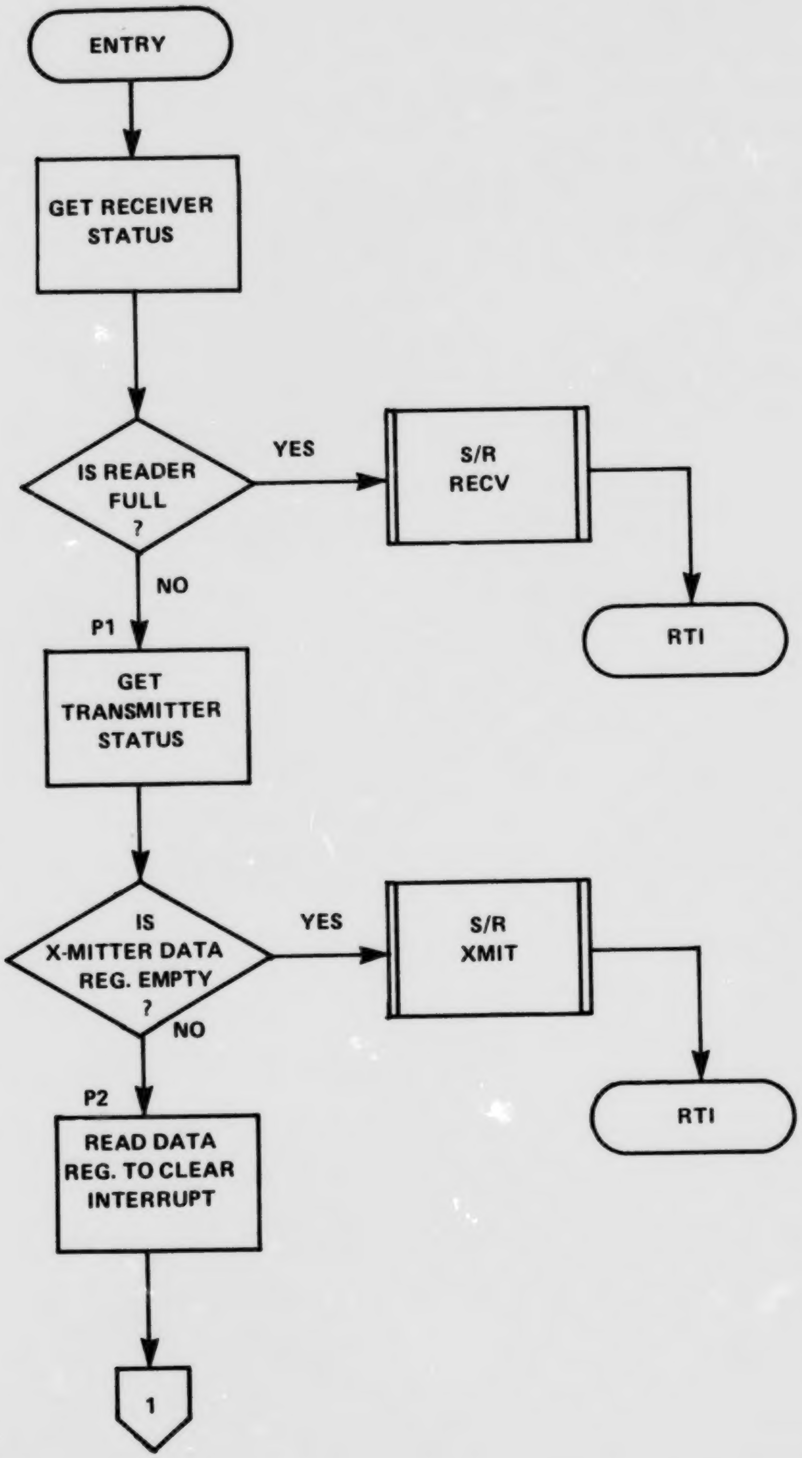
The receiver status (i.e., the ACIA status register) is examined. If a valid character was received then COMM invokes the receive routine RECV and returns from the interrupt.

Otherwise, the transmitter status bit is examined and, if the transmit register is empty (meaning a byte has been successfully sent), then the transmit routine (XMIT) is invoked and COMM then returns from the interrupt.

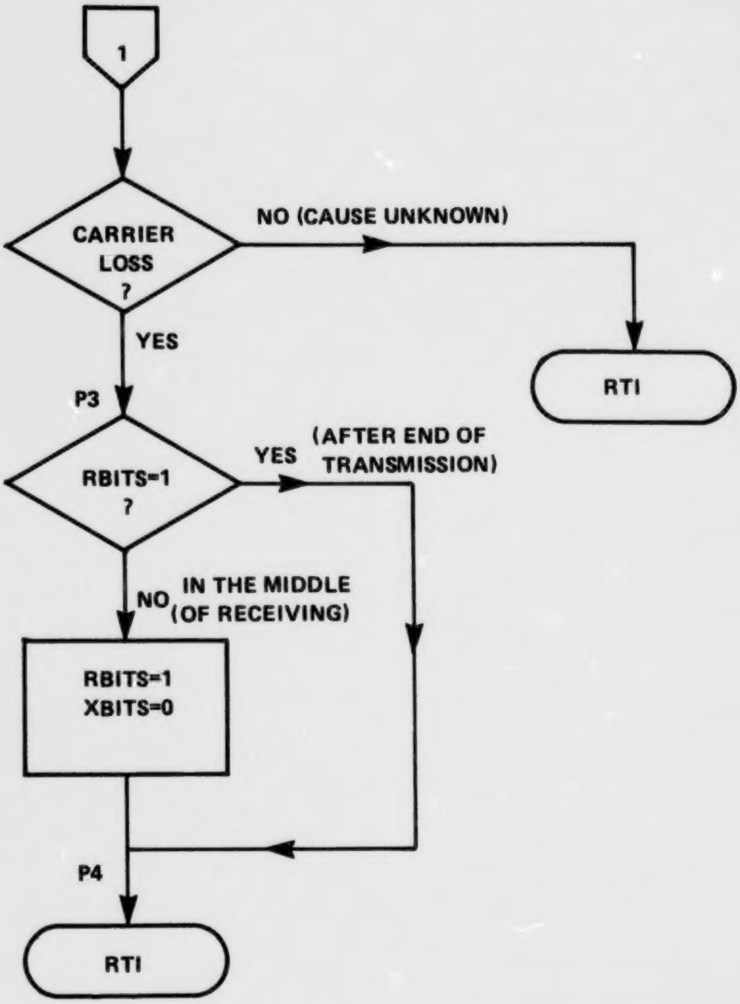
If neither a successful transmission nor a successful receipt caused the interrupt, the interrupt is cleared. If the interrupt was caused by a data carrier loss, then the controller is reset to receive mode (only if carrier loss occurred in the middle of receiving) and the routine terminates. If the interrupt was not caused by carrier loss, then the source of interrupt is unknown and the routine terminates.

Note -- Normally the controller is in receive state (RBITS = 1, XBITS = 0), since it cannot initiate a transmission except in response to a message from the central computer. This state is ensured on power-up (via routine POWRON) and is switched to a transmit state only after a message has been received. The reply is then sent out, and the controller once again enters the receive state. In the event of communication difficulties (line failure before entire message received or sent), the communication timer routines TIMER1 and TIMER2 will return the controller to the receiving state so the next message can be received.

COMMUNICATIONS PROGRAM – MAIN ROUTINE (COMM) Page 1of 2


ENTRY
GET RECEIVER STATUS
IS READER FULL ?
YES
S/R RECV
RTI
NO
P1
GET TRANSMITTER STATUS
IS X-MITTER DATA REG. EMPTY ?
YES
S/R XMIT
RTI
NO
P2
READ DATA REG. TO CLEAR INTERRUPT
1

COMMUNICATIONS PROGRAM – MAIN ROUTINE (COMM) Page 2 of 2


1
CARRIER LOSS ?
NO (CAUSE UNKNOWN)
RTI
YES
P3
RBITS=1 ?
YES (AFTER END OF TRANSMISSION)
NO (IN THE MIDDLE OF RECEIVING)
RBITS=1
XBITS=0
P4
RTI

7.6.2/ Receiving Routine (RECV)

Purpose

This subroutine is invoked from COMM whenever a character has been received over the communications port. It checks for input errors and, if an error has occurred, an appropriate response is prepared. Note that only one character is processed on any call to RECV.

If there are no input errors, RECV will process the character on the basis of the current receiving state. For example, if the first byte of the message is expected, RECV will check for a correct station address. Thus, the processing done depends on how much of the message has already been read. In all cases, however, the receiving routine will enter a new state after the input has been successfully processed (usually the state indicator byte, RBITS, is incremented by 1). In this new state the controller waits for the next character unless the end of the message has been encountered. Any error will cause a negative acknowledgement message to (eventually) be sent to the central computer.

When the end of the message is detected, the reply is prepared and sent (via PREANS and TFIRST).

Description

The format of communication messages is described in Appendix A. The received character is read from the ACIA. It is checked for errors (parity, overrun or framing). If there are any errors, the following is done.

- If the beginning of the message was expected (RBITS = 1) then start the receive timer. Otherwise, increment the error count (errors before the start of a message are probably spurious errors caused by closing the line after transmission of the previous message).

- Using EOMAF, prepare for the end of the message unless it has already been done for this message (i.e., unless RBITS is greater than 7).

In what follows, a valid character has been received. The processing depends on how much of the message we have already read, as indicated by RBITS.

I/ RBITS = 1 -- beginning of message expected

The receive deadlock timer CTIME1 is started. The station address (extracted from the first byte of the message) is compared to the correct station address. If they do not match, an error message is prepared by calling EOMAF and the routine terminates. Otherwise, a message checksum is initiated and the transaction code (extracted from the first message byte) is saved (in TCODE) for use by subroutine PREANS. The receiving routine then enters the next state (RBITS = 2) and the routine terminates.

II/ RBITS = 2 -- number of bytes to checksum expected

This number must be between 4 and 22, inclusive. If it is not, an error message is prepared (via EOMAF) and the routine terminates. If the number is valid the checksum is updated, the length of the message is computed and saved, and an input buffer is set up to receive the body of the message. Finally, the next receive state (RBITS = 3) is entered and the routine terminates.

III/ RBITS = 3 -- message text byte expected

The message byte is saved in an input buffer (addressed by DPOINT) and the checksum is updated. The remaining length of the message text is decremented. If this is the last byte, the next receive state (RBITS = 4) is entered and the routine terminates. Otherwise, the input buffer pointer is set to the next empty location and the routine terminates.

IV/ RBITS = 4 -- checksum byte expected

The received checksum is compared to the calculated checksum. If they are not equal, an error message is prepared (via EOMAF) and the routine terminates. Otherwise, a logical message, complete except for the end characters, has been received. Thus, subroutine PREANS is invoked to prepare an answer message. The next receive state (RBITS = 5) is then entered and the routine terminates.

V/ RBITS = 5 -- first end character expected

The received character is compared to $AA, the first end character. If they are not equal, the communications error count CERROR is incremented.

The next receive state (RBITS = 6) is then entered and the routine terminates.

VI/ RBITS = 6 -- second end character expected

The received character is compared to $55, the correct second end character. If they are not equal, the communication error count CERROR is incremented.

The next receive state (RBITS = 7) is then entered and the routine terminates.

VII/ RBITS = 7 -- third end character expected

The received character is compared to $55, the third end character. If they are not equal, the error count CERROR is incremented.

The entire message has now been received. The subroutine TFIRST is called to initiate transmission of the reply prepared by PREANS when the checksum byte was encountered. The routine then terminates.

* * * * *

The following states occur only after an error has been detected and EOMAF has been invoked. These three states are designed to find the end of the message by searching for the end sequence.

VIII/ RBITS = 8 -- expecting first end character after failure

If the character is $AA, the receiver enters the next state (RBITS = 9). The routine then terminates.

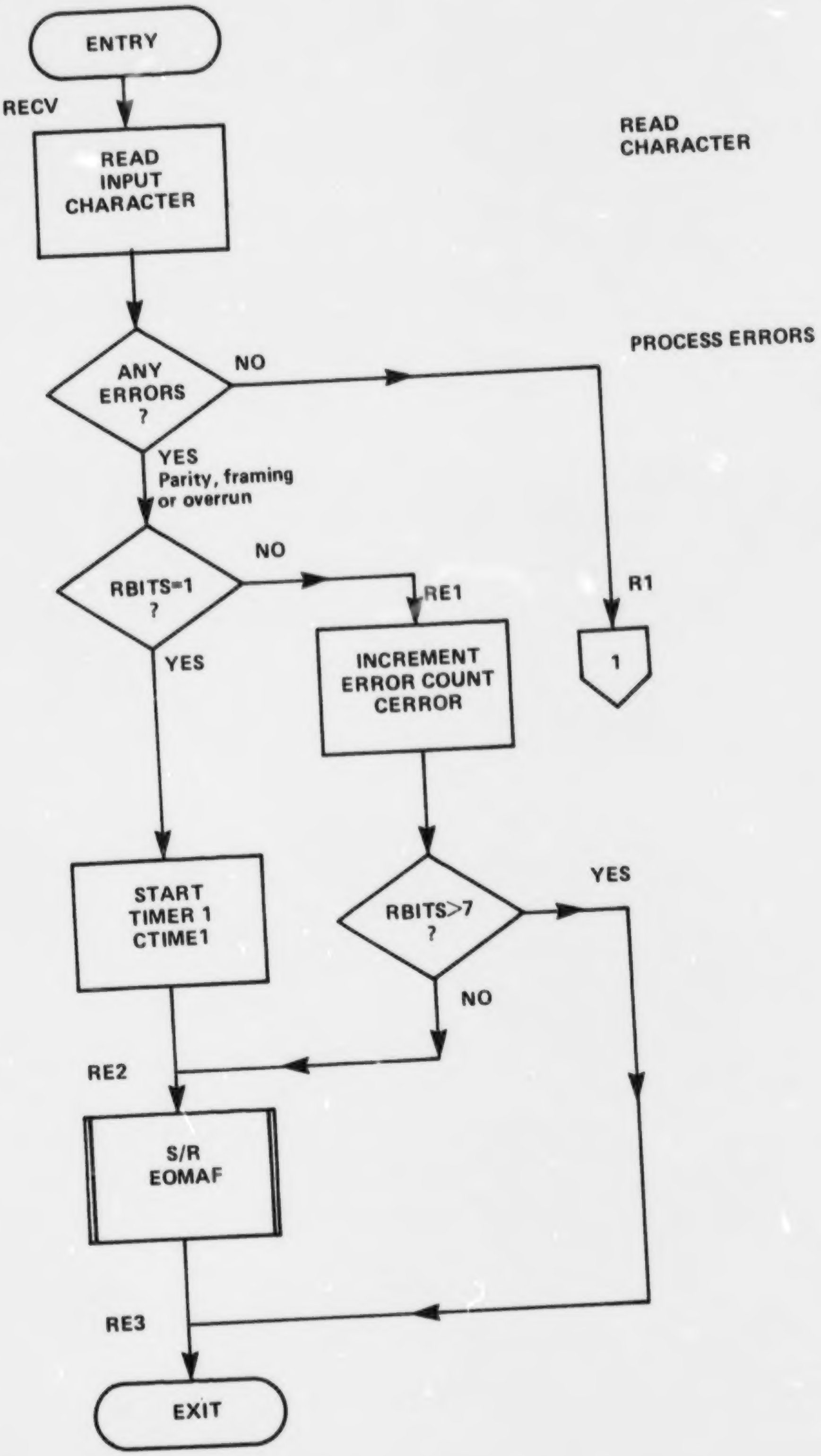
RBITS = 9 -- expecting second end character after failure

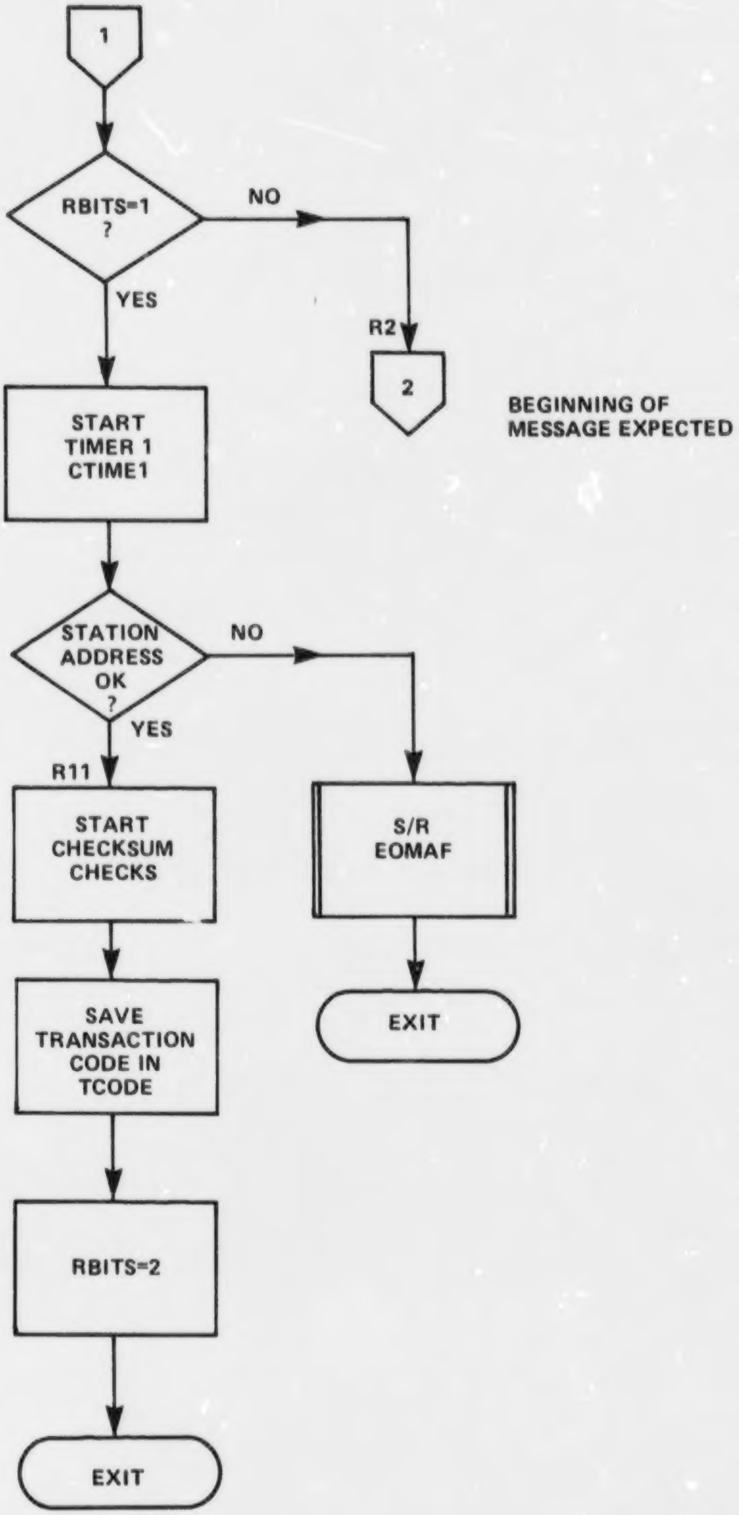
If the received character is $55, the receiver enters the next state (RBITS = 10) and the routine terminates. Otherwise, the state "expecting first end character after failure" is re-entered (RBITS = 8) and the routine terminates.

X/ RBITS = 10 -- expecting third end character after failure

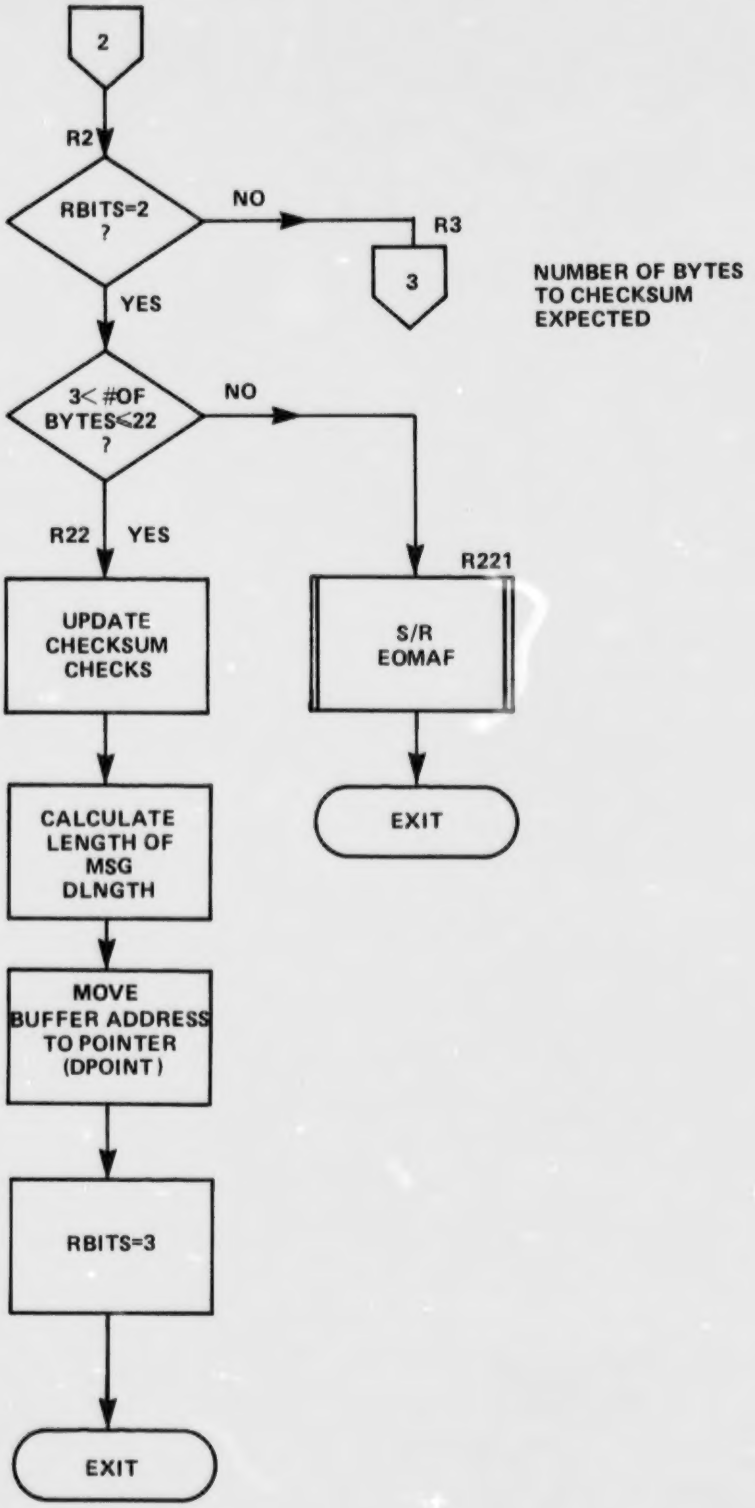
If the character is $55, TFIRST is invoked to initiate transmission of the negative acknowledgement message prepared by subroutine EOMAF. Otherwise, the receiver re-enters the "expecting first end character after failure" state (RBITS = 8). The routine then terminates.

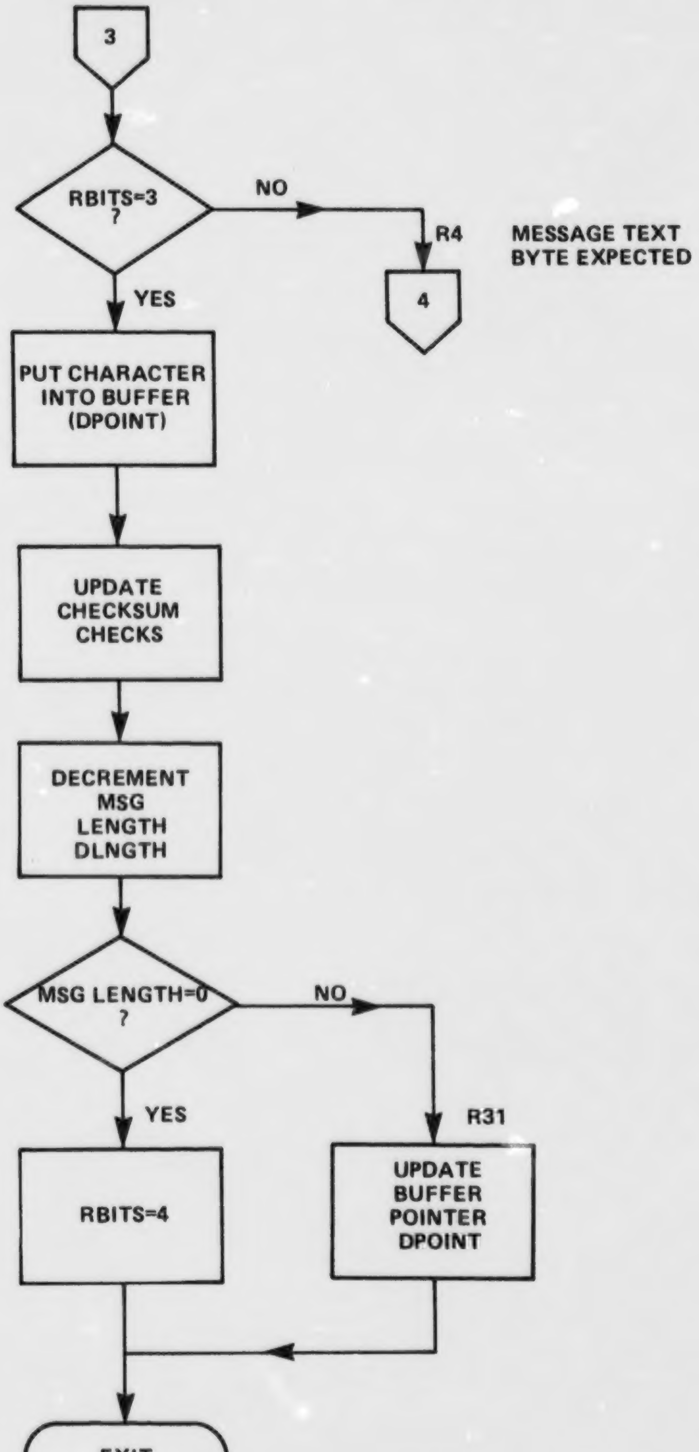
RECEIVE ROUTINE (RECV)

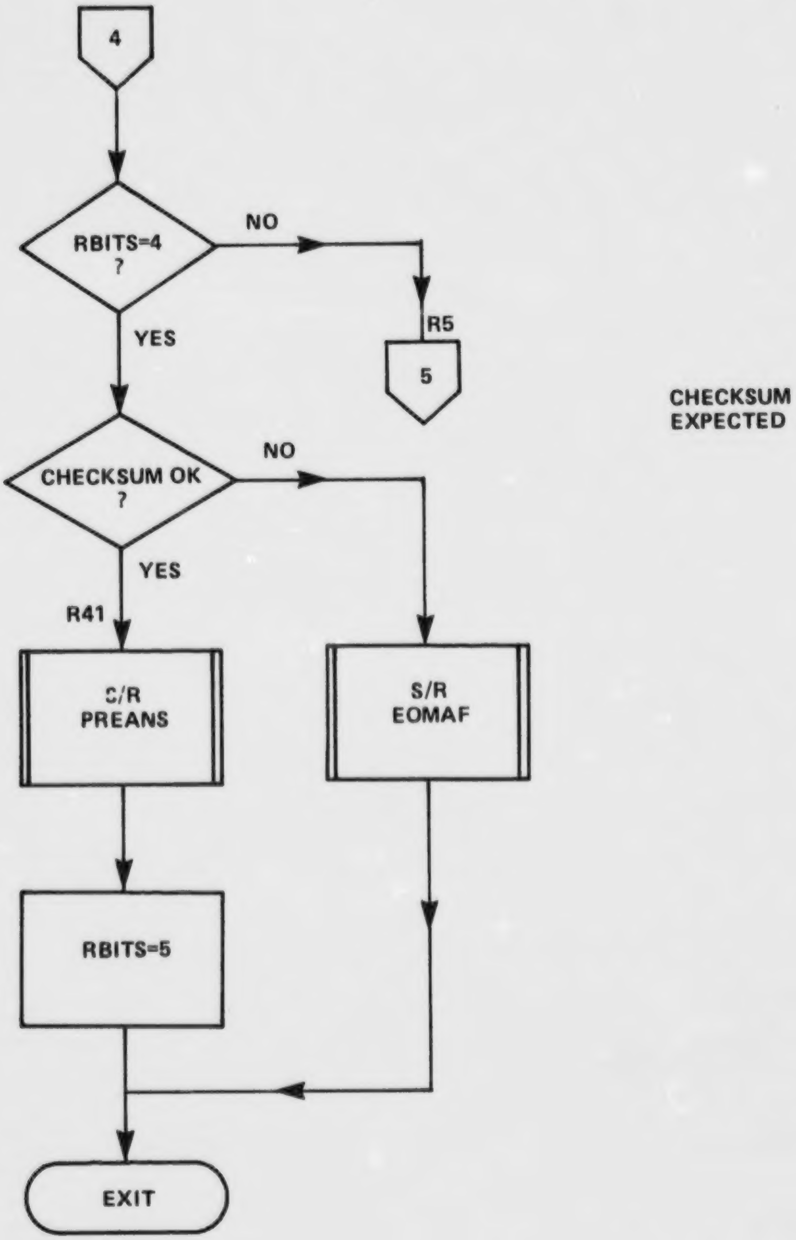

ENTRY
RECV
READ INPUT CHARACTER
READ CHARACTER
ANY ERRORS ?
NO
PROCESS ERRORS
YES
Parity, framing or overrun
RBITS=1 ?
NO
RE1
R1
YES
INCREMENT ERROR COUNT CERROR
1
START TIMER 1 CTIME1
RBITS>7 ?
YES
NO
RE2
S/R EOMAF
RE3
EXIT


1
RBITS=1 ?
NO
YES
R2
2
BEGINNING OF MESSAGE EXPECTED
START TIMER 1 CTIME1
STATION ADDRESS OK ?
NO
YES
R11
START CHECKSUM CHECKS
S/R EOMAF
EXIT
SAVE TRANSACTION CODE IN TCODE
RBITS=2
EXIT

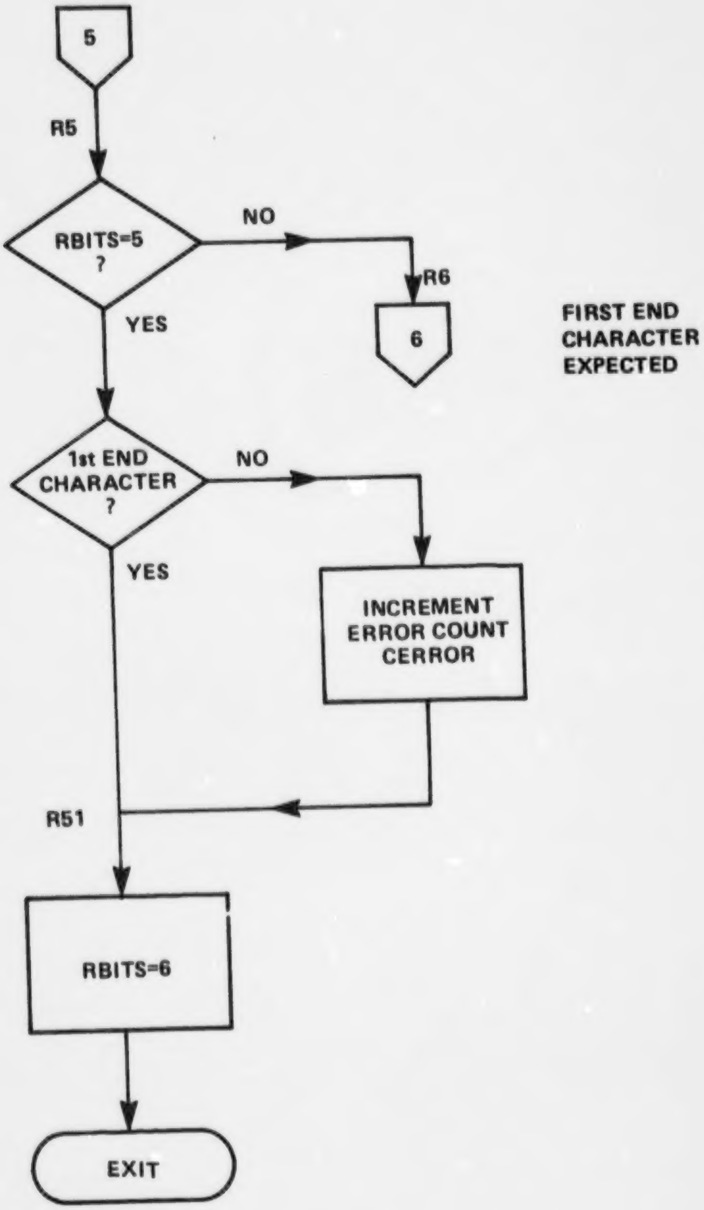
RECEIVE ROUTINE (RECV) Page 3 of 11


2
R2
RBITS=2 ?
NO
R3
3
NUMBER OF BYTES TO CHECKSUM EXPECTED
YES
3< #OF BYTES≤22 ?
NO
R22
YES
R221
UPDATE CHECKSUM CHECKS
S/R EOMAF
CALCULATE LENGTH OF MSG DLNGTH
EXIT
MOVE BUFFER ADDRESS TO POINTER (DPOINT)
RBITS=3
EXIT

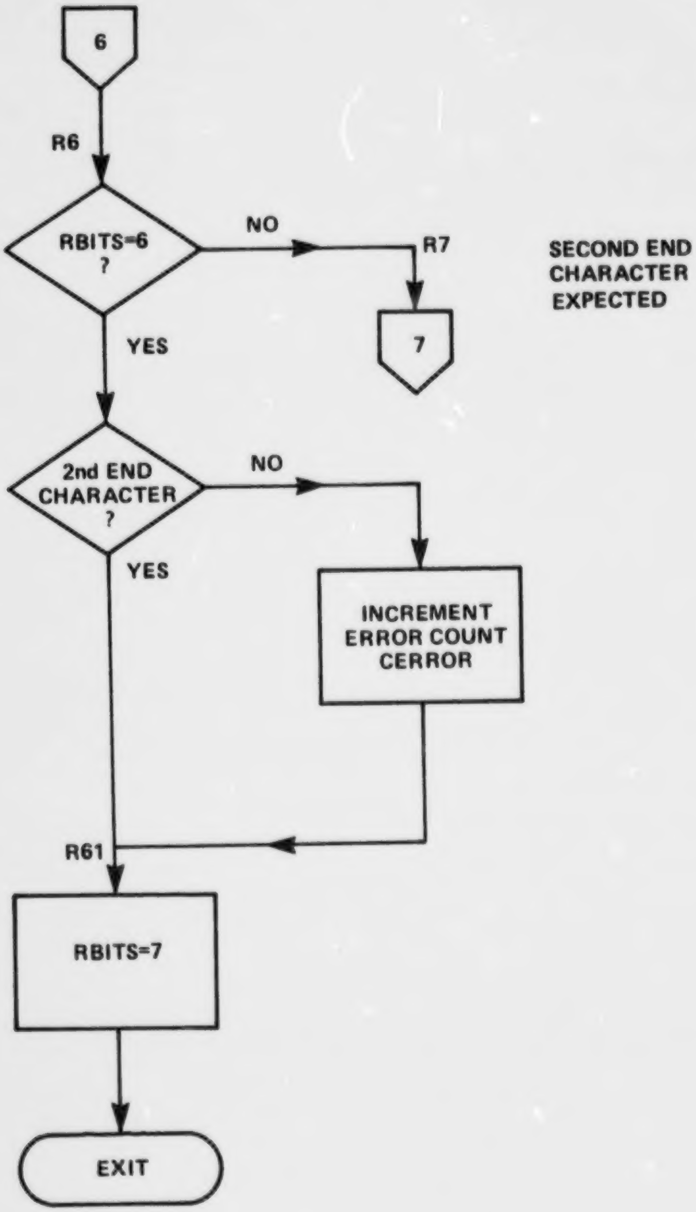

3
RBITS=3 ?
NO
R4
MESSAGE TEXT BYTE EXPECTED
4
YES
PUT CHARACTER INTO BUFFER (DPOINT)
UPDATE CHECKSUM CHECKS
DECREMENT MSG LENGTH DLNGTH
MSG LENGTH=0 ?
NO
YES
R31
RBITS=4
UPDATE BUFFER POINTER DPOINT
EXIT


4
RBITS=4 ?
NO
R5
5
YES
CHECKSUM EXPECTED
CHECKSUM OK ?
NO
YES
R41
S/R PREANS
S/R EOMAF
RBITS=5
EXIT

RECEIVE ROUTINE (RECV)

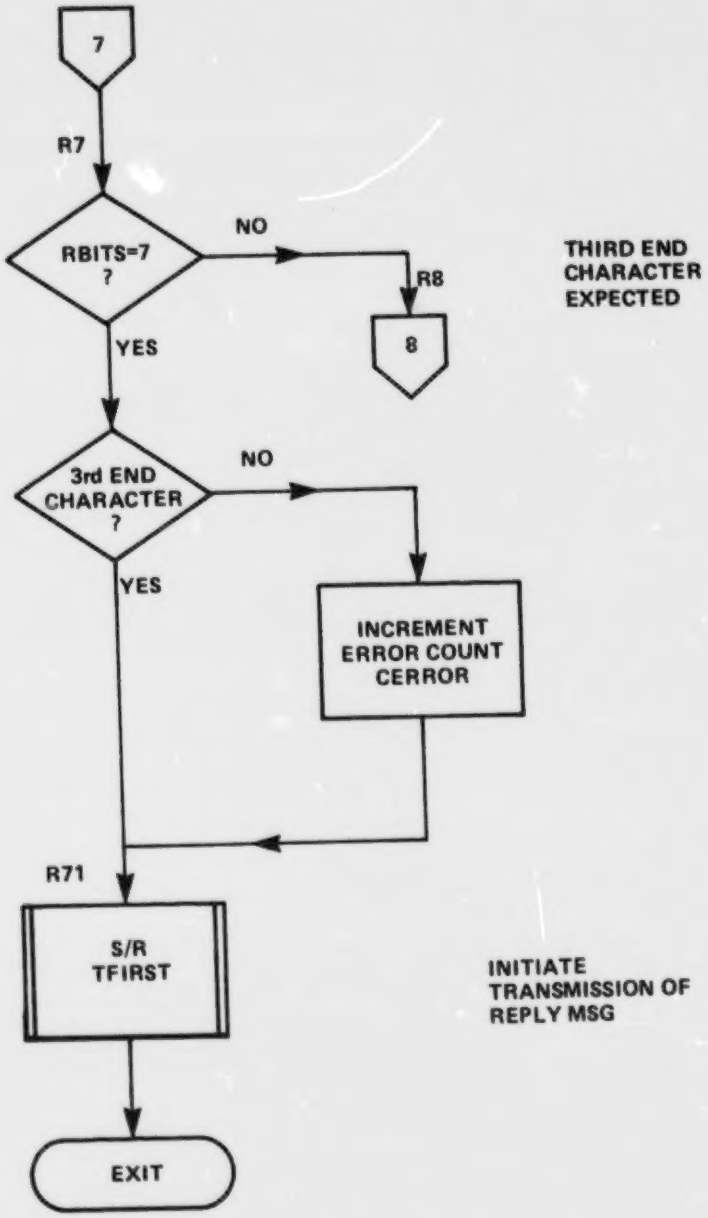

5
R5
RBITS=5
?
NO
R6
6
FIRST END
CHARACTER
EXPECTED
YES
1st END
CHARACTER
?
NO
YES
INCREMENT
ERROR COUNT
CERROR
R51
RBITS=6
EXIT

RECEIVE ROUTINE (RECV)


6
R6
RBITS=6
?
NO
R7
SECOND END
CHARACTER
EXPECTED
7
YES
2nd END
CHARACTER
?
NO
YES
INCREMENT
ERROR COUNT
CERROR
R61
RBITS=7
EXIT

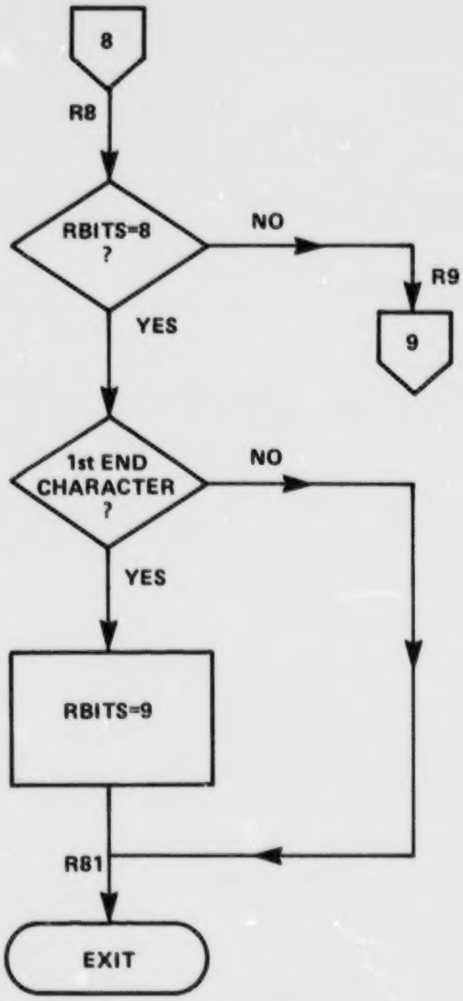
RECEIVE ROUTINE (RECV)

Page 8 of 11

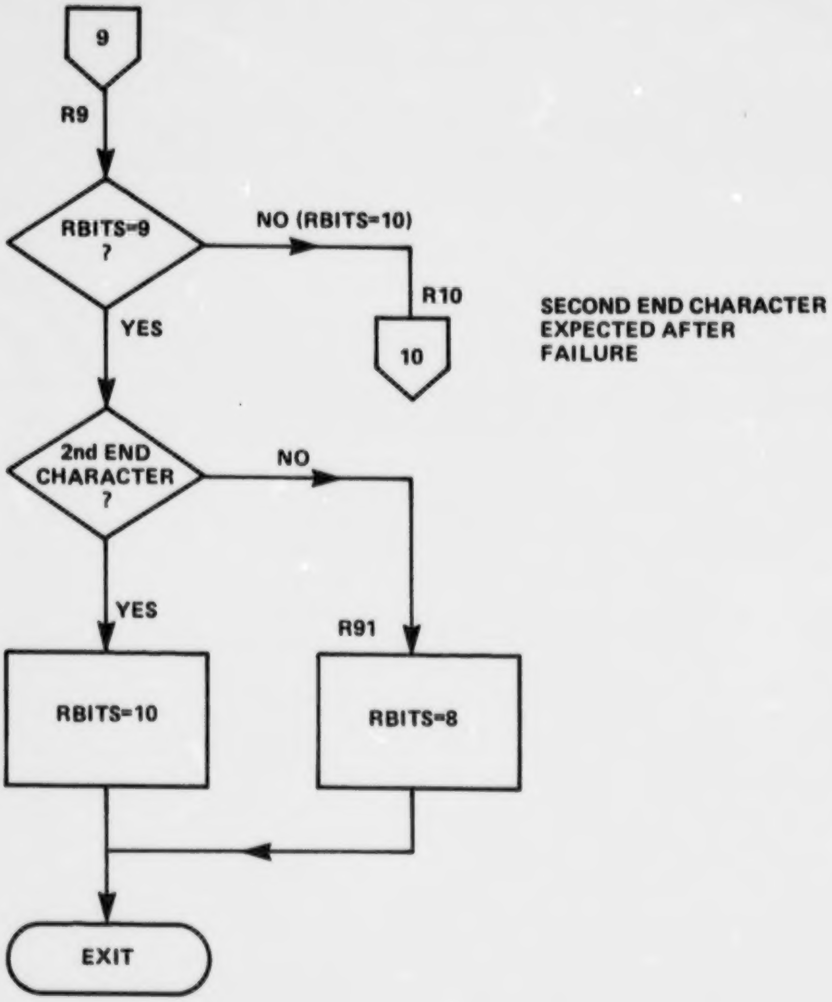

7
R7
RBITS=7
?
NO
YES
R8
8
THIRD END
CHARACTER
EXPECTED
3rd END
CHARACTER
?
NO
YES
INCREMENT
ERROR COUNT
CERROR
R71
S/R
TFIRST
INITIATE
TRANSMISSION OF
REPLY MSG
EXIT

RECEIVE ROUTINE (RECV)

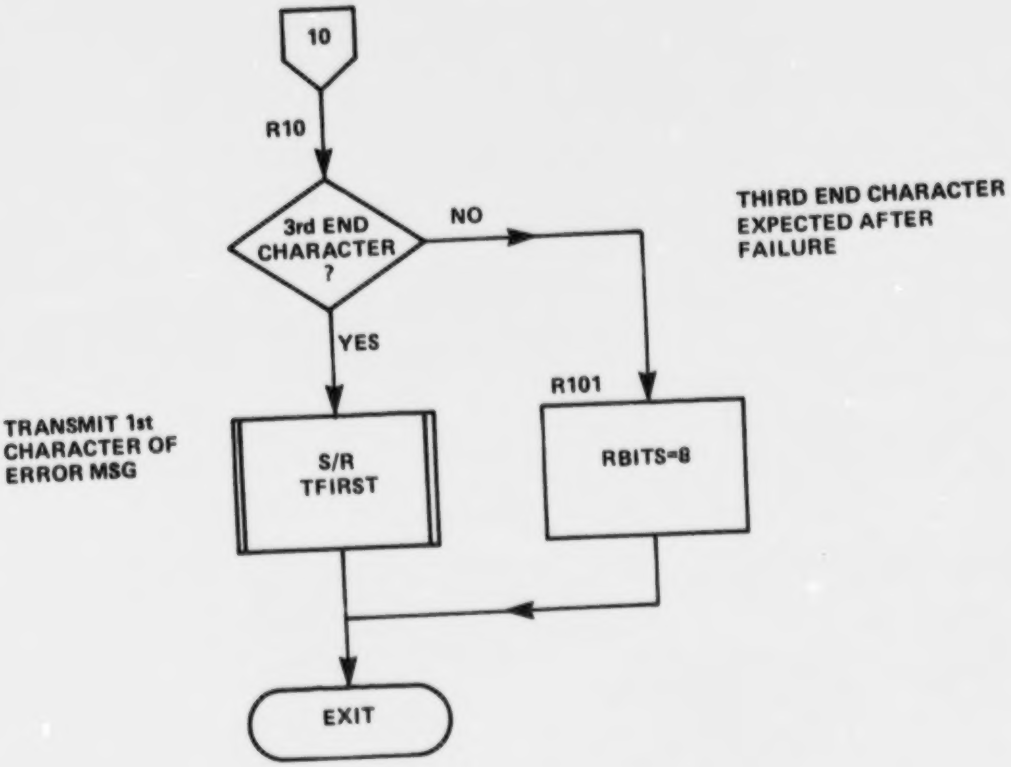
Page 9 of 11


8
R8
RBITS=8 ?
NO
R9
9
YES
1st END CHARACTER ?
NO
YES
RBITS=9
R81
EXIT

SEARCHING FOR FIRST END CHARACTER AFTER FAILURE


9
R9
RBITS=9
?
NO (RBITS=10)
R10
10
SECOND END CHARACTER
EXPECTED AFTER
FAILURE
YES
2nd END
CHARACTER
?
NO
YES
R91
RBITS=10
RBITS=8
EXIT

RECEIVE ROUTINE (RECV)


10
R10
3rd END CHARACTER ?
NO
YES
THIRD END CHARACTER EXPECTED AFTER FAILURE
R101
TRANSMIT 1st CHARACTER OF ERROR MSG
S/R TFIRST
RBITS=8
EXIT

7.6.2.1/ Prepare for End of Message after Failure Subroutine (EOMAF)

Purpose

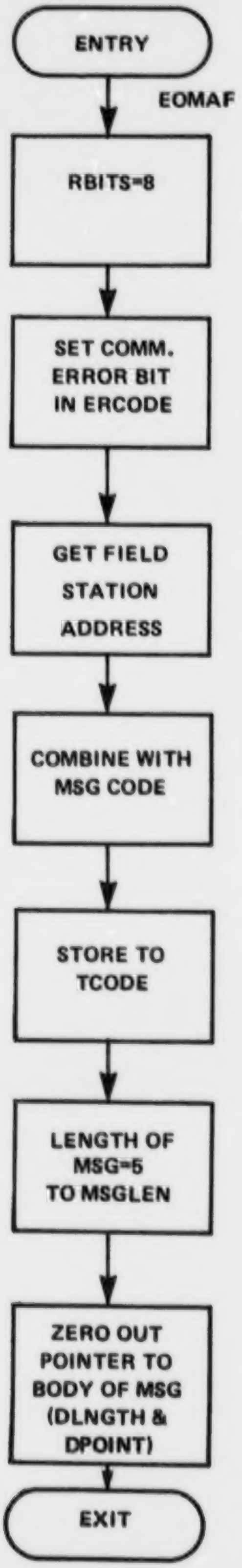
This subroutine is called by RECV. During receipt of a communication message, if an error was detected in the message (invalid checksum, invalid byte count, invalid station address, or some hardware error), this routine is invoked. It puts the receiving routine in a state where it attempts to find the end of the message. It then prepares a negative acknowledgement message for transmission.

Description

First, the "waiting for end of message after failure" state (RBITS = 8) is entered. The communication error bit of ERCODE is then set. Finally, a negative acknowledgement response is prepared for transmission.

PREPARE FOR END OF MSG AFTER FAILURE S/R (EOMAF)

Page 1 of 1


ENTRY
EOMAF
RBITS=8
SET COMM. ERROR BIT IN ERCODE
GET FIELD STATION ADDRESS
COMBINE WITH MSG CODE
STORE TO TCODE
LENGTH OF MSG=5 TO MSGLEN
ZERO OUT POINTER TO BODY OF MSG (DLNGTH & DPOINT)
EXIT

PREPARE NEGATIVE ACKNOWLEDGEMENT MESSAGE

7.6.3/ Transmit First Character Routine (TFIRST)

Purpose

This routine turns the receiver off and then turns the transmitter on. The first character of the message to the central computer is then sent.

TFIRST is invoked when a message has been completely received or by the expiration of the receive timer (subroutine TIMER1).

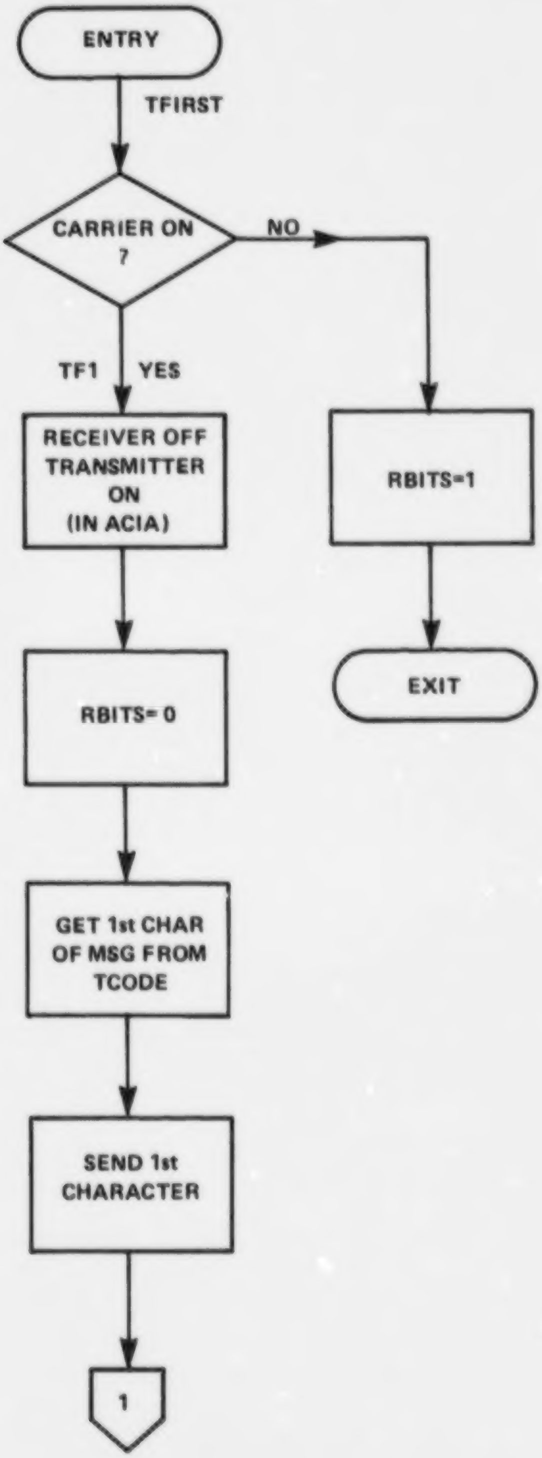
Description

First, the routine checks for the presence of the data carrier. If it is not present, receiving mode (RBITS = 1) is entered and the routine terminates.

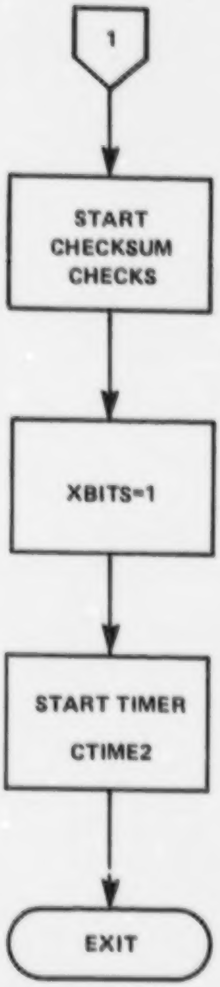
Otherwise, the receiver is turned off and the transmitter is turned on in both the hardware (ACIA) and software (RBITS = 0, XBITS = 1). The first character of the message is sent, the checksum is initialized and the transmit timer (CTIME2) is started.

TRANSMIT 1st CHARACTER OF MSG ROUTINE (TFIRST)

Page 1 of 2


ENTRY
TFIRST
CARRIER ON ?
NO
TF1
YES
RECEIVER OFF TRANSMITTER ON (IN ACIA)
RBITS=1
EXIT
RBITS= 0
GET 1st CHAR OF MSG FROM TCODE
SEND 1st CHARACTER
1

TRANSMIT 1st CHARACTER OF MSG POUTINE (TFIRST) Page 2 of 2


1
START
CHECKSUM
CHECKS
XBITS=1
START TIMER
CTIME2
EXIT

7.6.4/ Transmit Routine (XMIT)

Purpose

This routine is called by COMM whenever a character has been successfully transmitted over the communication line. It either sends the next character of the message to the central computer or it switches to receiving mode when the entire message has been transmitted.

The processing performed depends on the current transmitting state, as indicated by XBITS, which depends on how much of the message has already been sent (the format of communication messages is described in Appendix A). After the required processing is performed, the transmit routine will enter the next state.

Description

The current transmitting state (XBITS) is examined. If the complete message has been sent (XBITS = 10), or the data carrier has been lost, then the transmission is terminated and the receive state is entered (XBITS = 0, RBITS = 1), the hardware is prepared to receive, and then the routine terminates.

Otherwise, the transmit timer CTIME2 is restarted. This timer is used to detect line failure in the transmitting state, since the data carrier interrupt is disabled while transmitting. The subsequent processing depends on how much of the message has already been sent, as indicated by XBITS. The various cases are discussed below.

I/ XBITS = 1 -- send 1st byte of message length

The first byte of the message length (MSGLEN) is sent, the checksum is updated and the next transmit state (XBITS = 2) is entered. The routine then terminates.

II/ XBITS = 2 -- send 2nd byte of message length

The second byte of the message length (MSGLEN+1) is sent, the checksum is updated and the next transmit state (XBITS = 3) is entered. The routine then terminates.

III/ XBITS = 3 -- send a byte from data buffer (if applicable)

If there is nothing left to be sent (DLNGTH = 0), then proceed as in case IV, below.

Otherwise, the next byte from the data buffer is sent, the checksum is updated and the number of bytes yet to be sent (DLNGTH) is decremented. If the buffer is now empty (DLNGTH = 0), then the second buffer is checked. If it too is empty, then the next transmit state (XBITS = 4) is entered and the routine terminates. If the second buffer is not empty, then it is moved to the first buffer and the second buffer is then cleared (actually, only pointers are moved and cleared).

If the first buffer is not empty, the next byte to be sent is addressed.

The routine then terminates.

IV/ XBITS = 4 -- send error code byte

The error code (ERCODE) is transmitted, the checksum is updated and the next transmit state (XBITS = 5) is entered. The routine then terminates.

V/ XBITS = 5 -- send checksum byte

The checksum byte is sent, the next transmit state is entered (XBITS = 6) and the routine terminates.

VI/ XBITS = 6 -- send 1st end character

The character $AA is sent, the next transmit state (XBITS = 7) is entered and the routine terminates.

VII/ XBITS = 7 -- send 2nd end character

The character $55 is sent, the next transmit state (XBITS = 8) is entered and the routine terminates.

VIII/ XBITS = 8 -- send 3rd end character

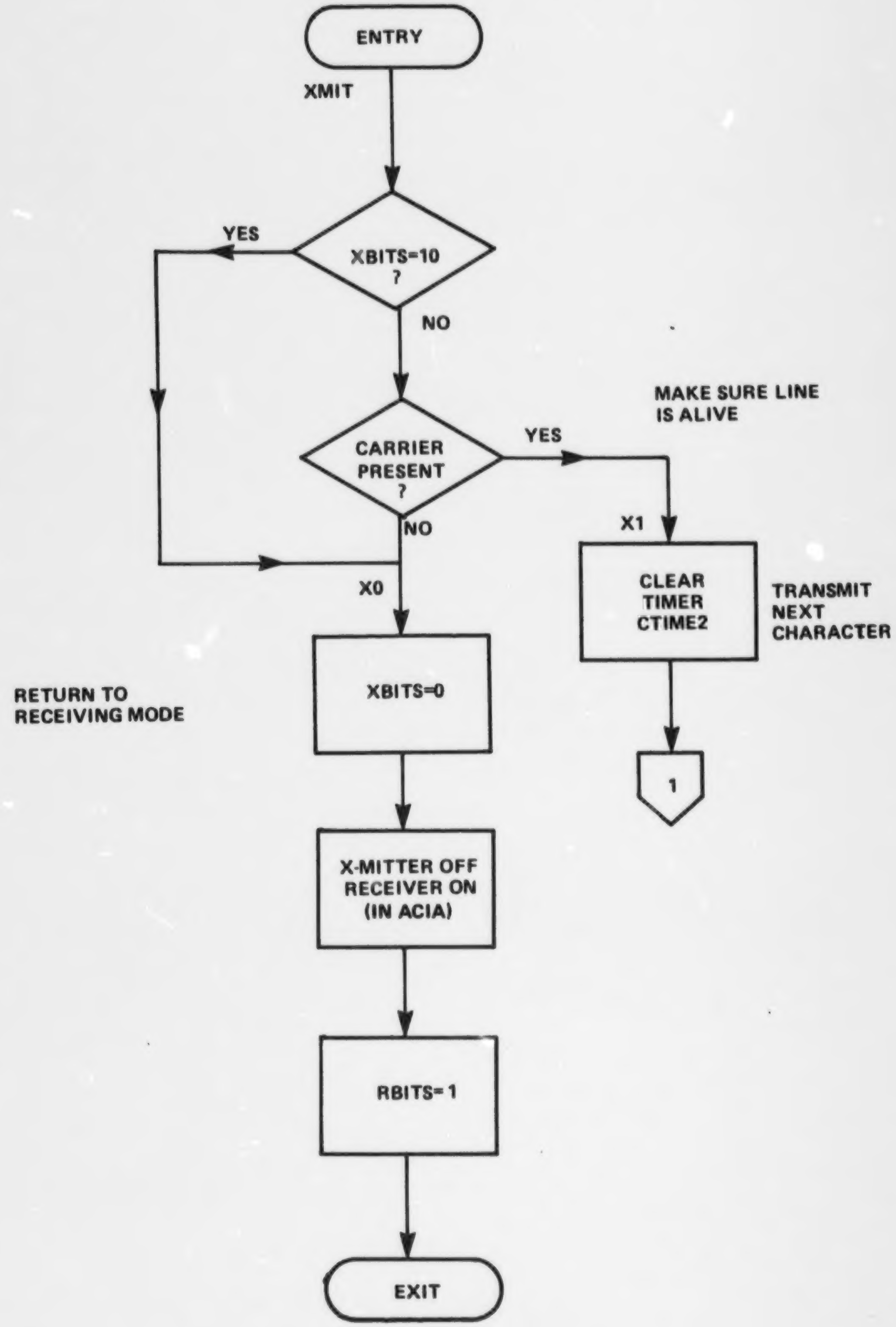
The character $55 is sent, the next transmit state (XBITS = 9) is entered and the routine terminates.

IX/ XBITS = 9 -- send 4th end character

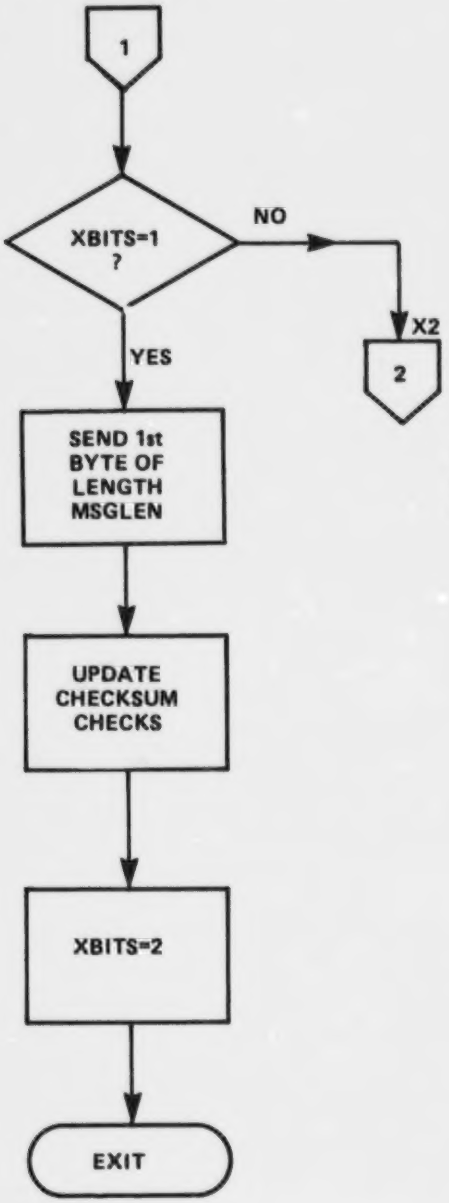
The character $55 is sent, the next transmit state (XBITS = 10) is entered and the routine terminates.

TRANSIT ROUTINE (XMIT)

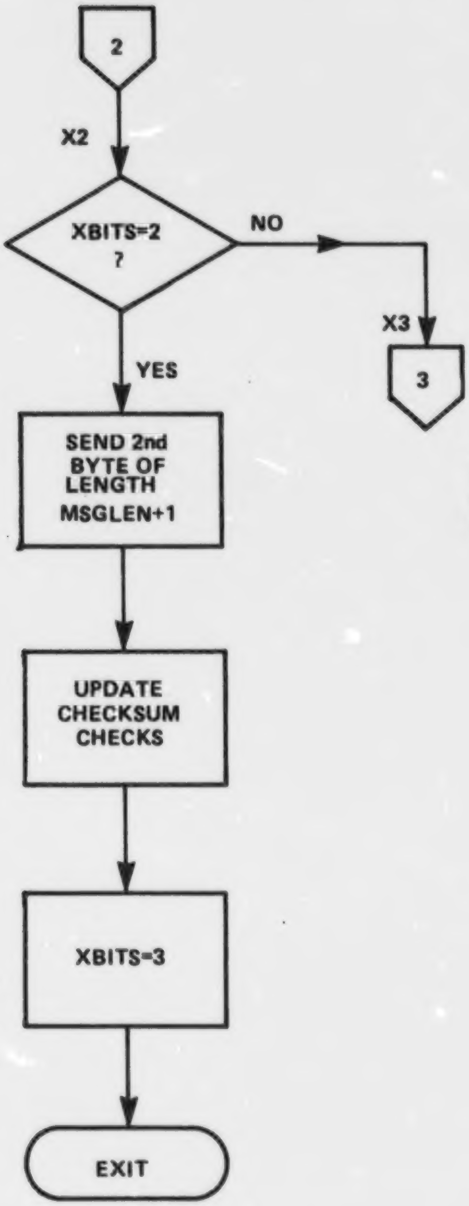
Page 1 of 9


ENTRY
XMIT
XBITS=10 ?
YES
NO
MAKE SURE LINE IS ALIVE
CARRIER PRESENT ?
YES
NO
X1
X0
CLEAR TIMER CTIME2
TRANSMIT NEXT CHARACTER
RETURN TO RECEIVING MODE
XBITS=0
1
X-MITTER OFF RECEIVER ON (IN ACIA)
RBITS=1
EXIT

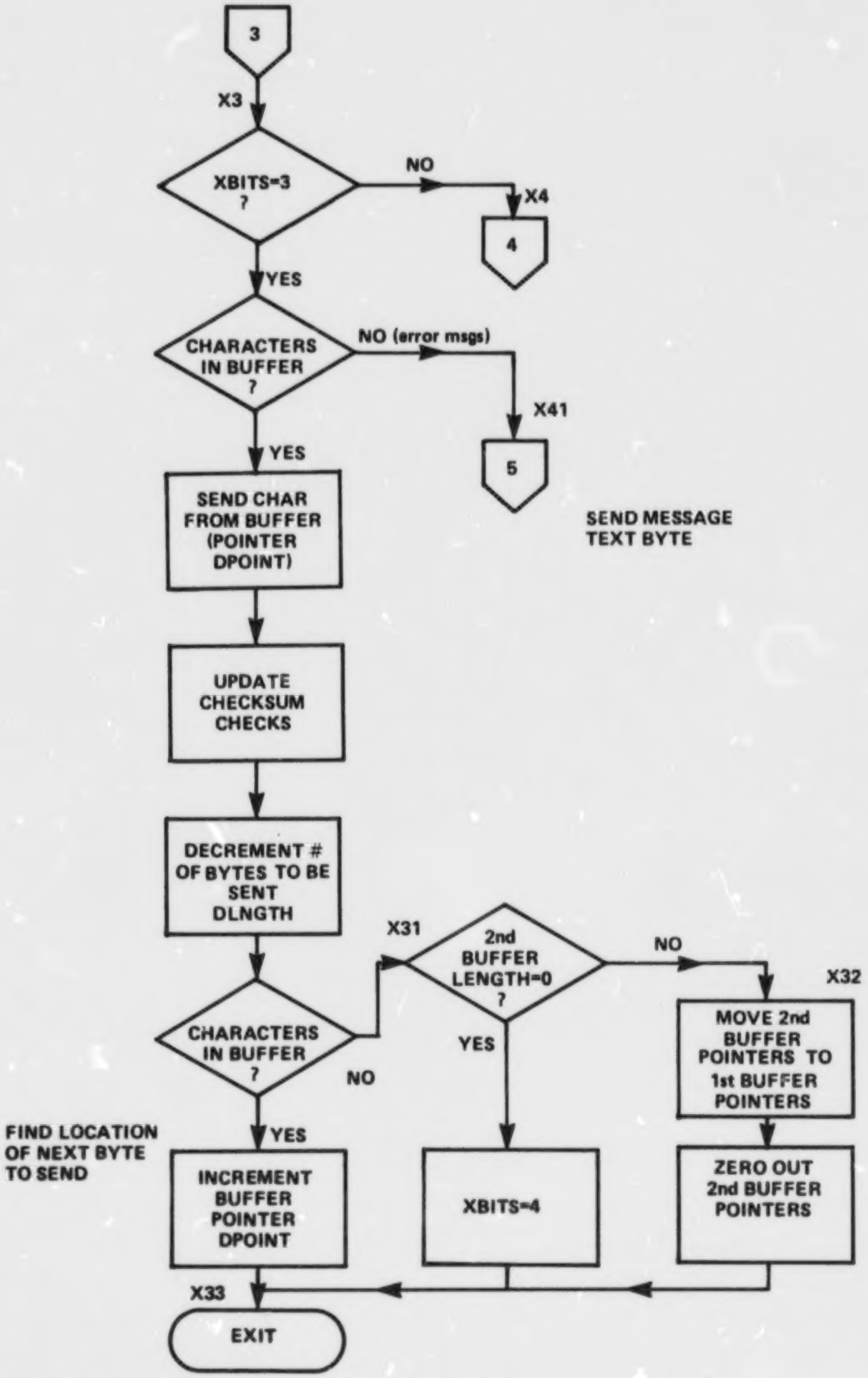
TRANSMIT ROUTINE (XMIT)


1
XBITS=1 ?
NO
X2
2
YES
SEND 1st BYTE OF LENGTH MSGLEN
UPDATE CHECKSUM CHECKS
XBITS=2
EXIT

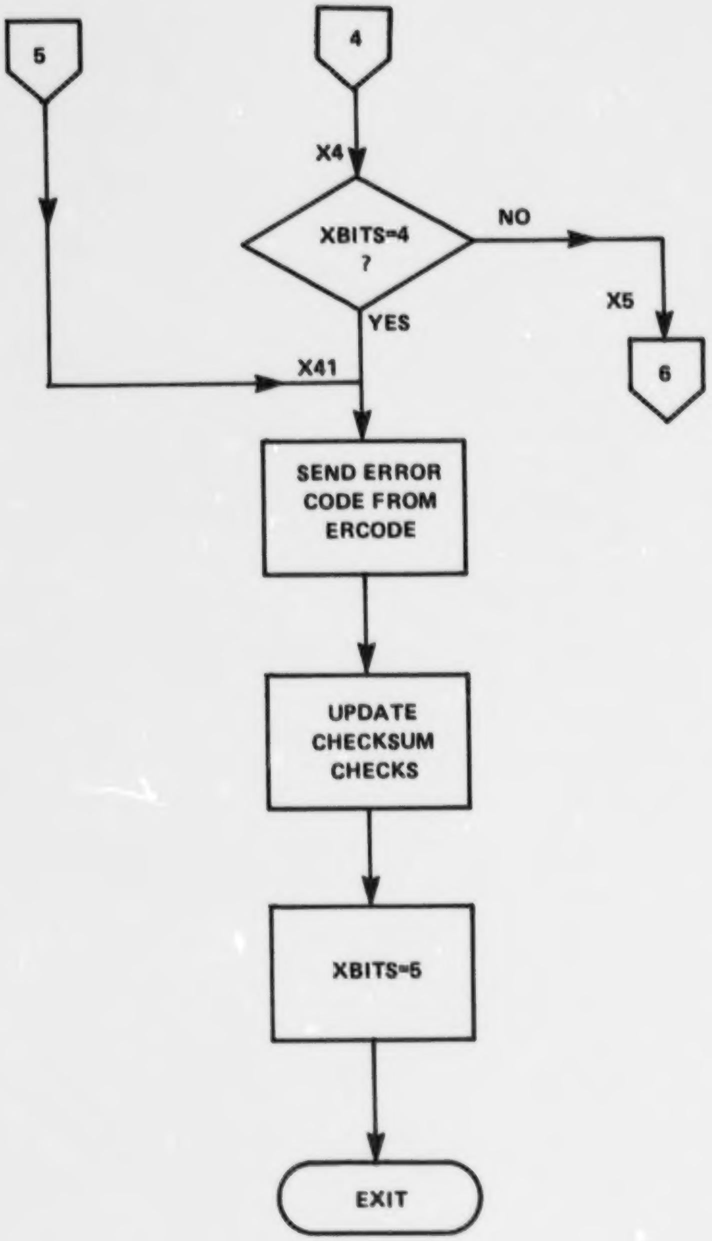
SEND FIRST BYTE OF MESSAGE LENGTH


2
X2
XBITS=2
?
NO
X3
3
YES
SEND 2nd BYTE OF LENGTH MSGLEN+1
UPDATE CHECKSUM CHECKS
XBITS=3
EXIT

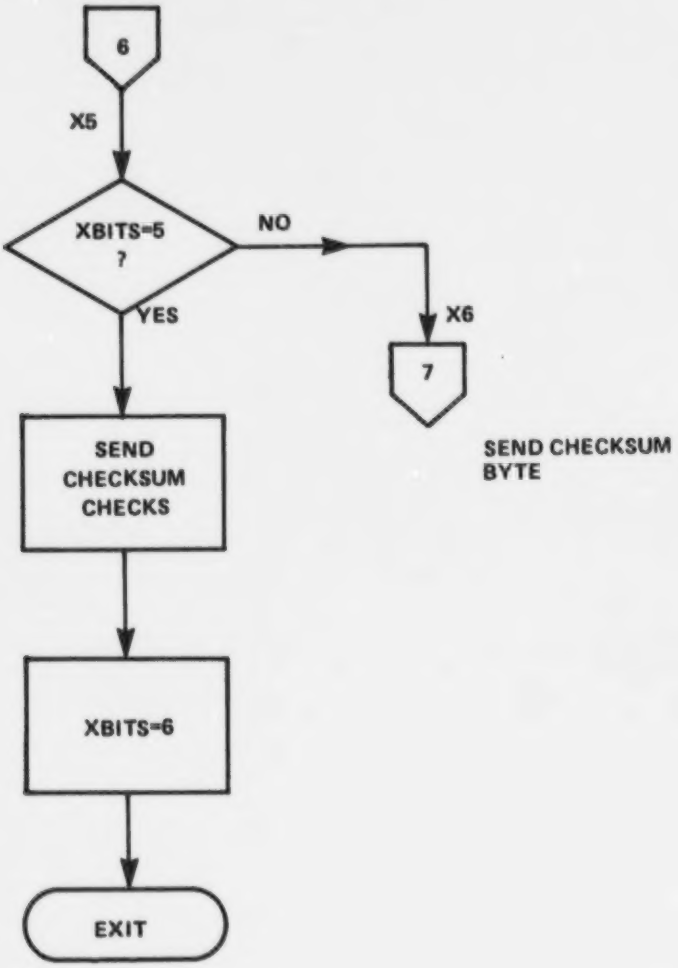
SEND SECOND CHARACTER OF MESSAGE LENGTH


3
X3
XBITS=3 ?
NO
X4
4
YES
CHARACTERS IN BUFFER ?
NO (error msgs)
X41
5
YES
SEND CHAR FROM BUFFER (POINTER DPOINT)
SEND MESSAGE TEXT BYTE
UPDATE CHECKSUM CHECKS
DECREMENT # OF BYTES TO BE SENT DLNGTH
X31
2nd BUFFER LENGTH=0 ?
NO
X32
CHARACTERS IN BUFFER ?
NO
YES
MOVE 2nd BUFFER POINTERS TO 1st BUFFER POINTERS
FIND LOCATION OF NEXT BYTE TO SEND
YES
INCREMENT BUFFER POINTER DPOINT
XBITS=4
ZERO OUT 2nd BUFFER POINTERS
X33
EXIT

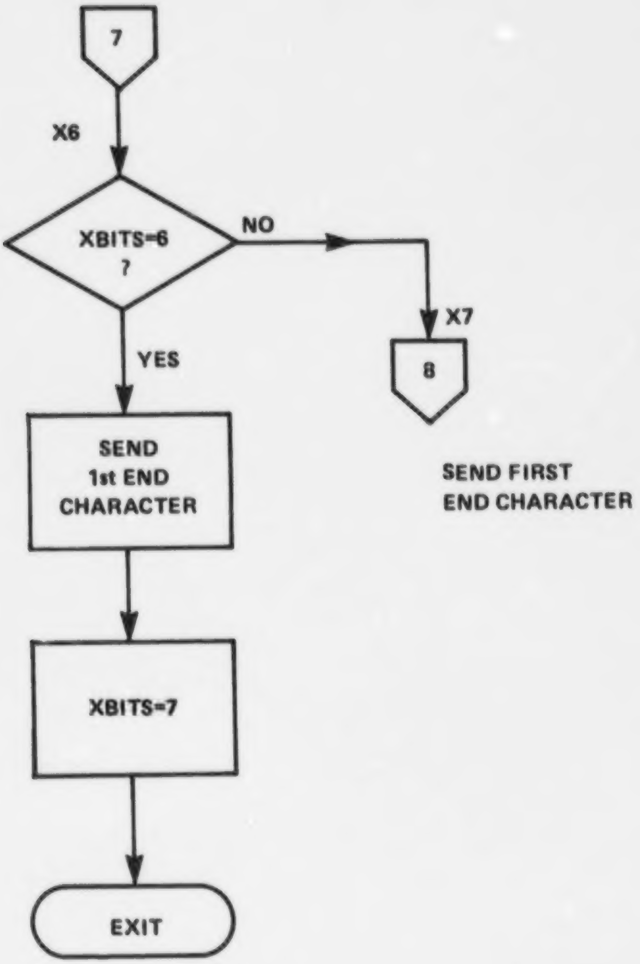
TRANSMIT ROUTINE (XMIT)

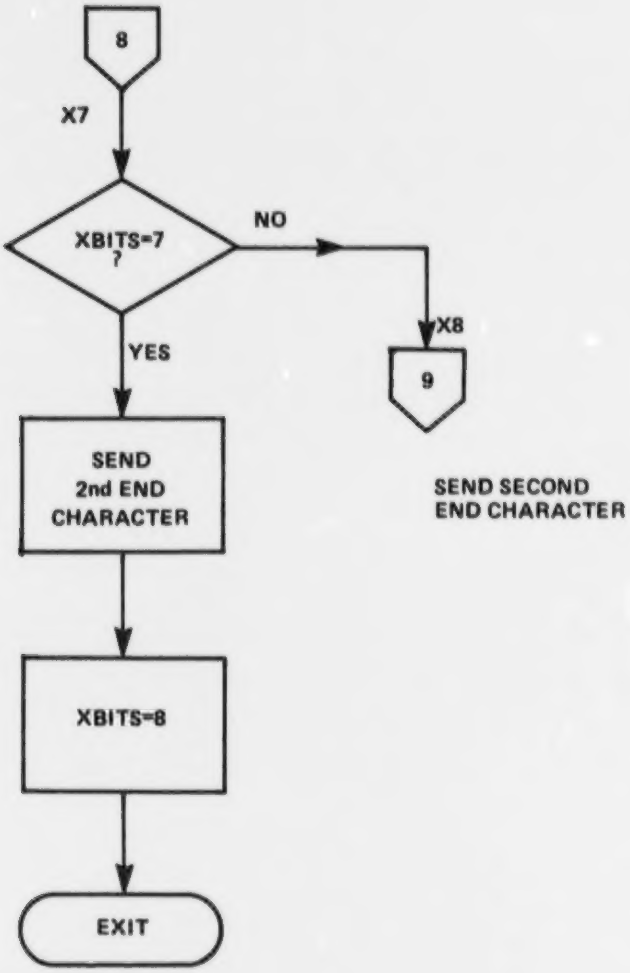

5
4
X4
XBITS=4
?
NO
X5
6
YES
X41
SEND ERROR
CODE FROM
ERCODE
UPDATE
CHECKSUM
CHECKS
XBITS=5
EXIT

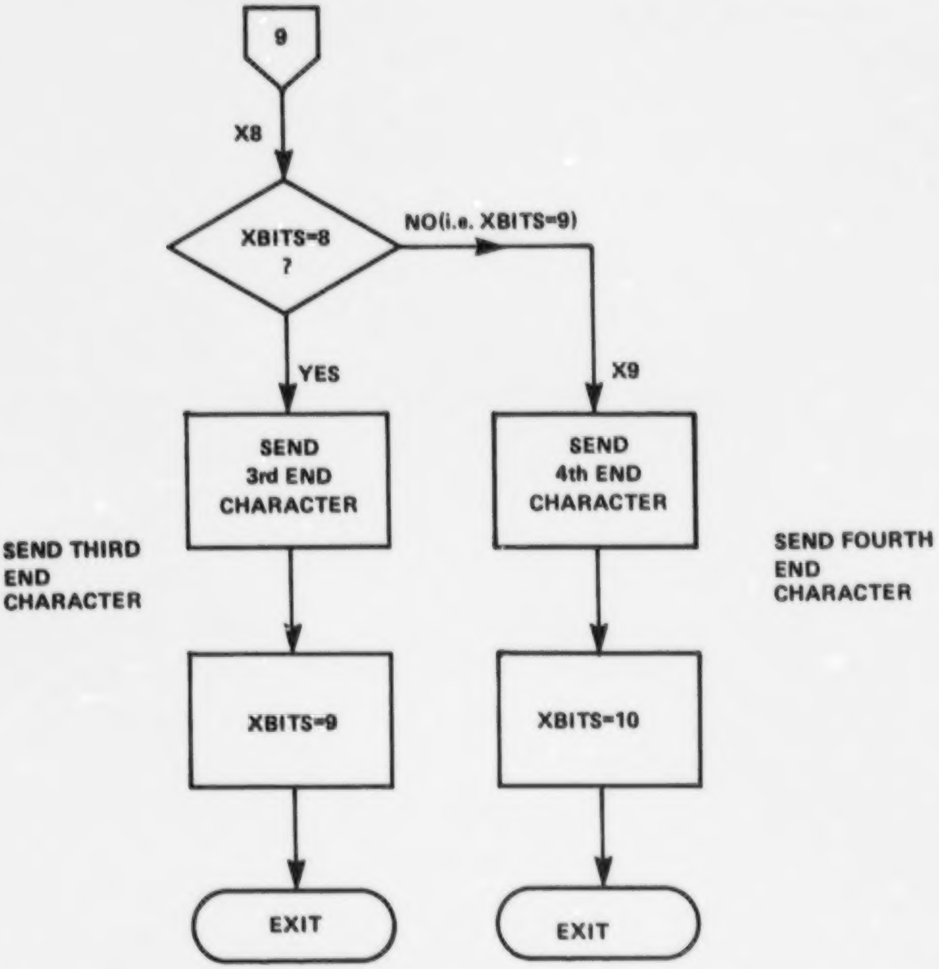
SEND ERROR
CODE BYTE


6
X5
XBITS=5
?
NO
YES
X6
7
SEND CHECKSUM BYTE
SEND
CHECKSUM
CHECKS
XBITS=6
EXIT

TRANSMIT ROUTINE (XMIT)


7
X6
XBITS=6
?
NO
YES
X7
8
SEND
1st END
CHARACTER
SEND FIRST
END CHARACTER
XBITS=7
EXIT


8
X7
XBITS=7
?
NO
X8
9
YES
SEND
2nd END
CHARACTER
SEND SECOND
END CHARACTER
XBITS=8
EXIT


9
X8
XBITS=8
?
NO(i.e. XBITS=9)
YES
X9
SEND
3rd END
CHARACTER
SEND
4th END
CHARACTER
SEND THIRD
END
CHARACTER
SEND FOURTH
END
CHARACTER
XBITS=9
XBITS=10
EXIT
EXIT

7.6.5/ Prepare Answer Routine (PREANS)

Purpose

The routine analyses a message once it has been received from the central computer, and then takes appropriate action. It is invoked by the receive routine RECV after the checksum byte of the message is received and found to be correct. An appropriate reply is prepared for transmission.

There are 8 possible message codes to and from the central computer (codes 000-111), though only 6 are allowed to be sent from the central computer (the other 2 result in a logical error message).

This routine first processes those elements common to all messages (for the format of these messages, refer to the program listing or Appendix A). Then, it prepares a response. Because there is no separate output buffer, the reply message is addressed by two sets of buffer access variables. The first set, DPOINT and DLNGTH, address the beginning of the message and its length. The second set, POINT2 and LNGTH2, are for the second part of the message if the reply message is in two separate parts in core. These variables are later used by the transmit subroutine when transmitting the body of the message.

NOTE -- this routine only prepares the reply message. The actual transmission occurs later when all the input message's end characters have been received or when the receive timer times out.

Description

Firstly, the message independent processing is done, that is, if there are any communication errors (i.e., CERROR is greater than 0) then the communication error bit of ERCODE is set. Otherwise, the bit is reset. Similarly, if there are any detector errors, the detector malfunction bit of ERCODE is set. Otherwise, it is reset. The message checksum is then cleared. The 3-bit message code (received from the central computer) is then examined. The processing performed depends on this message code and is outlined below.

I/ logical error message (000)

This message should only be sent by the controller, not by the central computer. However, if for some reason it is received by the controller, in response the routine prepares a logical error message (via subroutine LOGER) and terminates.

II/ send field station memory message (001)

The routine checks for the following conditions:
- incorrect message length,
- requested memory is partially outside of RAM,
- requested memory is partially outside of PROM,
- requested memory lies completely outside both RAM and PROM.

If any of these conditions exists, the routine invokes subroutine LOGER to prepare a logical error message and then terminates. Otherwise, the response message is prepared as follows:
- station address and message code (001) are set;
- length of message is set;
- first set of buffer access variables (DPOINT, DLNGTH) is set to starting address and number of bytes to send; and
- second set of buffer access variables (POINT2, LNGTH2) is set to transmit requested RAM or PROM.

The routine then terminates.

III/ negative acknowledgement message (010)

This message should never be received by the controller. If it is, subroutine LOGER is invoked to prepare a logical error message and the routine PREANS then terminates.

IV/ send hourly count volume data message (011)

This requests data for a particular day. The following conditions are checked:

- incorrect message length;
- data are not available for the requested date -- this may be caused by failure to perform master initialization, in which case pointers to data may be incorrect (the availability of data is determined by first ascertaining the validity of the pointers and then comparing the given date to those dates for which data are available, starting at VODAY1)

If any of these conditions exists, subroutine LOGER is invoked to prepare a logical error message and the routine terminates.

If the date is found, a reply message (011) is prepared for transmission, subroutine RESCON is invoked to process any applicable resets (time and detector errors), and the first set of buffer access variables is set to the beginning of the requested volume data and its length. The second buffer access variables are cleared.

The routine then terminates.

V/ send vehicle speed and length data message (100)

The following conditions are first checked:

- incorrect message length;
- either length or speed data are not available for the given date -- this may be caused by failure to perform master initialization, in which case the speed and length pointers may be incorrect (the availability of data is determined by first ascertaining the validity of the pointers and then comparing the given date to both those dates for which length data are available, starting at LNDAY1, and those for which speed data are available, starting at SPDAY1).

If either of these conditions exists, subroutine LOGER is invoked to prepare a logical error message and the routine terminates.

Otherwise, the first buffer access variables (DPOINT, DLNGTH) are set to reference the length data and the second buffer access variables are set to reference the speed data. A reply message (100) is then prepared in the usual manner. Also, subroutine RESCON is invoked to process any applicable resets. The routine then terminates.

VI/ start vehicle speed and length data (101) (not currently used)

The following conditions are checked:
- incorrect message length,
- date requested chronologically precedes the current date,
- the requested date is not valid (Feb 29 but not a leap year).

If any of these conditions exist, subroutine LOGER is invoked to prepare a logical error message and the routine terminates.
(Note -- invalid dates, such as March 32, will not be detected.)

Otherwise, a flag (in MFLAG) is set to indicate speed and length data are to be accumulated. The starting date is saved in SDATE and subroutine RESCON is called to process any resets. The response message (101) is then prepared for transmission in the usual manner (only the first set of buffer access variables is used, the second set is cleared).

The routine then terminates.

VII/ end vehicle speed and length count message (110)
(not currently used)

The following conditions are checked:
- incorrect message length;
- no speed data collected and start speed calculation bit of MFLAG not set (a "start speed and length data" message will set this bit);
- end date occurs before start date;
- end date occurs before current date;
- end date is invalid (Feb 29 but not a leap year -- but note that dates such as March 54 will not be detected as errors).

If any of these conditions exists subroutine LOGER is invoked to prepare a logical error message and the routine terminates.

Otherwise, the stop speed calculation bit of MFLAG is set, the end date is saved in EDATE and subroutine RESCON is invoked. The reply message (110) is prepared (clearing the second buffer access variables) and the routine terminates.

VIII/ change field station memory (111)

The following conditions are checked:

- message has incorrect length (maximum number of bytes changed = 16);
- number of changes received does not equal number stated in message;
- area to be changed (or part of it) lies outside of RAM:
- security code incorrect (not correct in core and not provided in message, or provided in message but incorrect).

Any of these conditions will cause a logical error message to be prepared (via LOGER) and termination of the routine.

Otherwise, the security code timer is started and the security code timer bit of MFLAG is set. The requested bytes are changed (except for protected locations) and an answer message (111) is prepared for transmission.

The address of the changed RAM and its length (as received in the message) are referenced by the first set of buffer access variables. The second set of buffer access variables are set to reference the RAM itself. The routine then terminates.

7.6.5.1/ Prepare Logical Error Message Subroutine (LOGER)

Purpose
This routine is called by PREANS and it prepares for the transmission of a logical error message (code 000) after PREANS has determined there is a logical error (as opposed to a hardware error) in the received message.

Description
The station address and the message code (000) are combined for transmission. The length of the message is set and both output buffer

access variables are cleared (since this message has no additional data). The routine then terminates.

7.6.5.2/ Reset Control Subroutine (RESCON)

Purpose

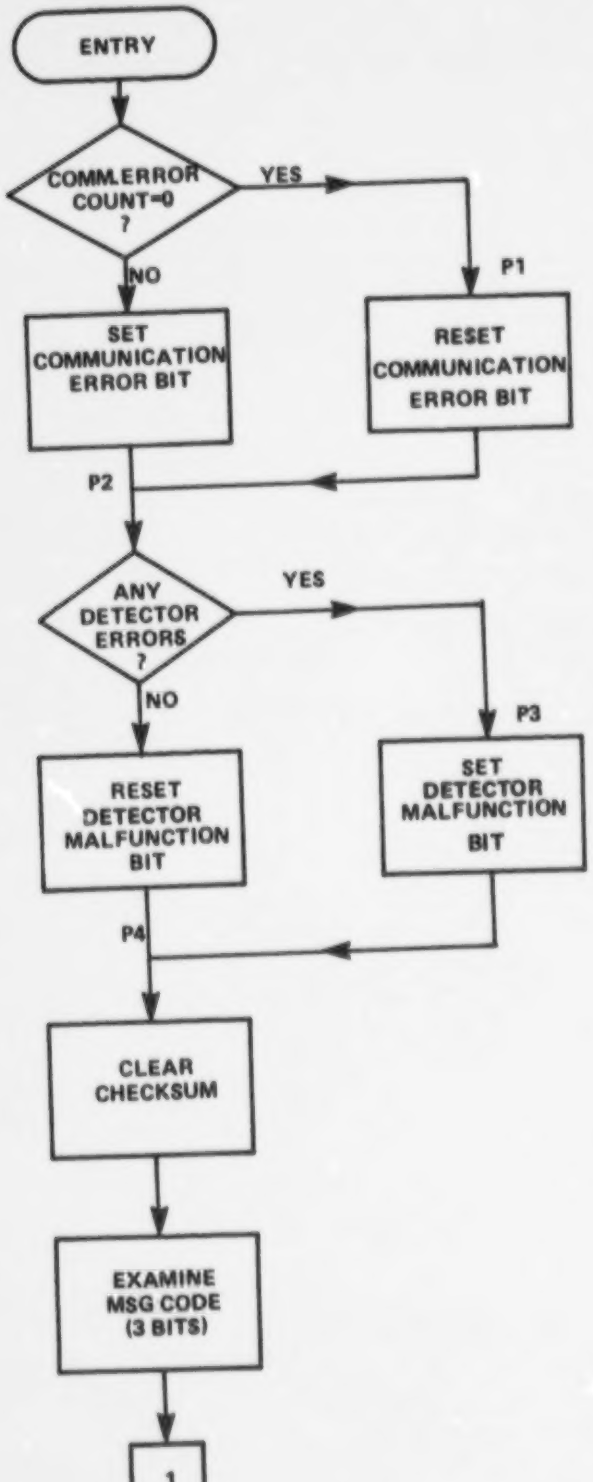
This subroutine is called by PREANS and it resets the detector error flags and the clock error flag, as indicated by the control byte of the message received from the central computer. Refer to the program listing for the meaning of the control byte.

Description

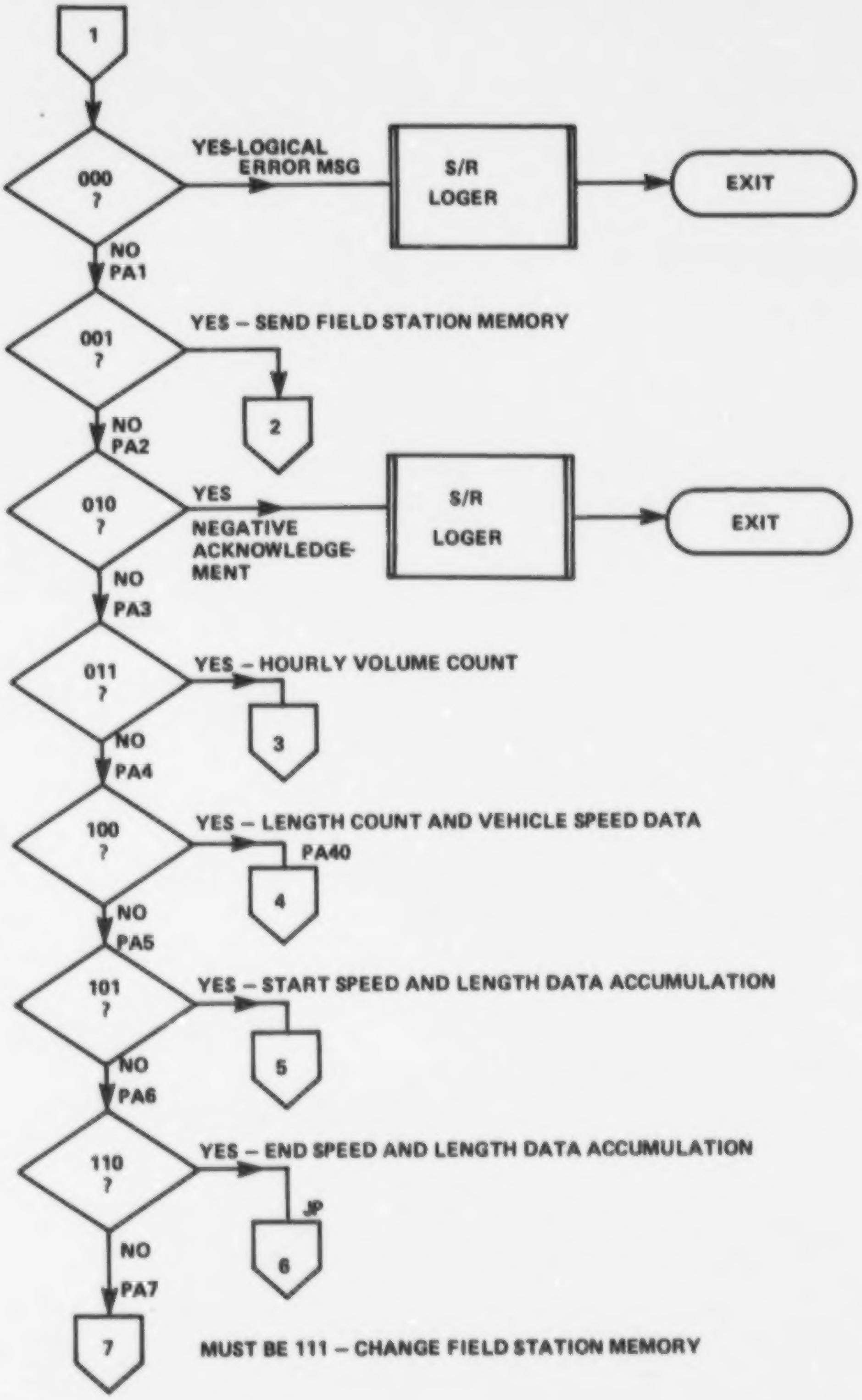
If the clear down-time bit of the control byte is set, the time-of-day clock error bit of ERCODE is reset and the power failure counter DWNCNT is cleared. If the clear detector error bit of the control byte is not set, the routine terminates. Otherwise, the lane identification to be reset is extracted from the control byte and the appropriate detector error bit (upstream or downstream, as determined from the control byte) is reset. Finally, if as a result of clearing this error bit there are no more detector errors, the detector error bit of ERCODE is reset.

Page 1 of 15

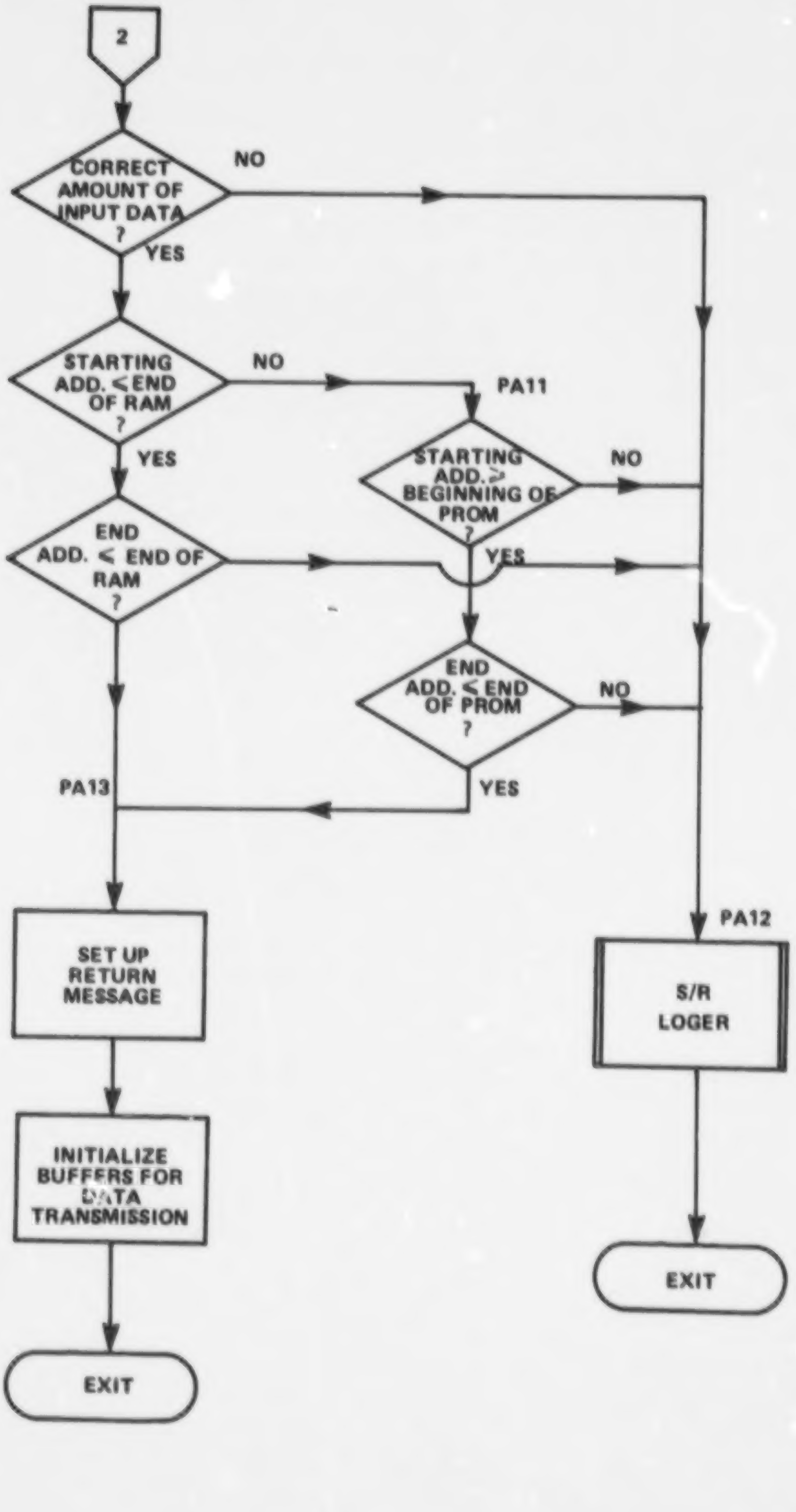
PREPARE ANSWER ROUTINE (PREANS)


ENTRY
COMM.ERROR COUNT=0 ?
YES
NO
P1
SET COMMUNICATION ERROR BIT
RESET COMMUNICATION ERROR BIT
P2
ANY DETECTOR ERRORS ?
YES
NO
P3
RESET DETECTOR MALFUNCTION BIT
SET DETECTOR MALFUNCTION BIT
P4
CLEAR CHECKSUM
EXAMINE MSG CODE (3 BITS)
1

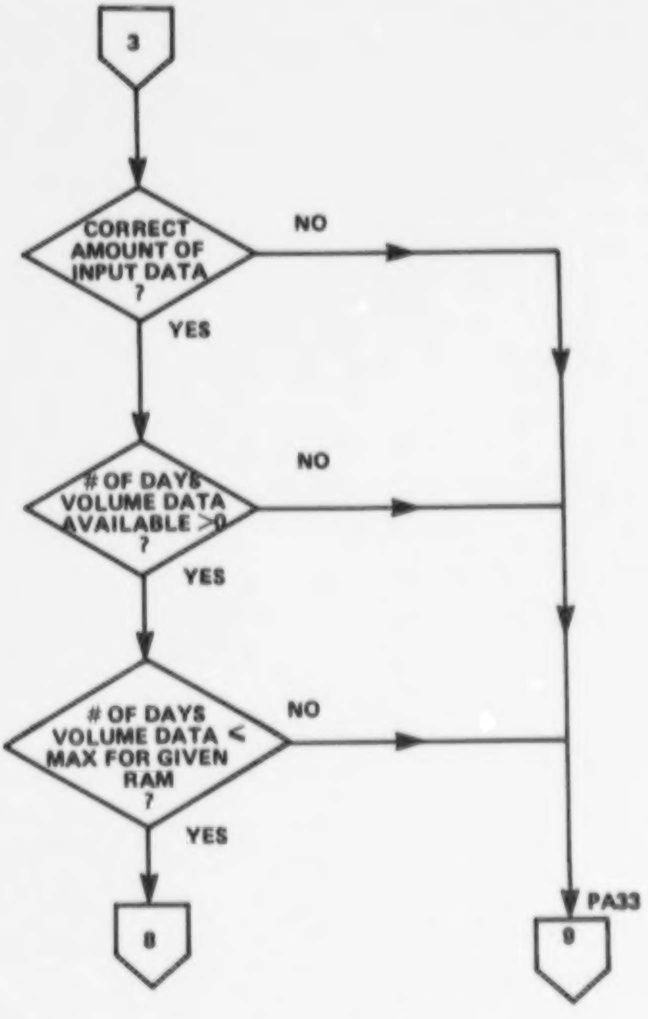
PREPARE ANSWER ROUTINE (PREANS)


1
000
?
YES-LOGICAL ERROR MSG
S/R
LOGER
EXIT
NO
PA1
001
?
YES – SEND FIELD STATION MEMORY
2
NO
PA2
010
?
YES
NEGATIVE ACKNOWLEDGEMENT
S/R
LOGER
EXIT
NO
PA3
011
?
YES – HOURLY VOLUME COUNT
3
NO
PA4
100
?
YES – LENGTH COUNT AND VEHICLE SPEED DATA
PA40
4
NO
PA5
101
?
YES – START SPEED AND LENGTH DATA ACCUMULATION
5
NO
PA6
110
?
YES – END SPEED AND LENGTH DATA ACCUMULATION
JP
6
NO
PA7
7
MUST BE 111 – CHANGE FIELD STATION MEMORY

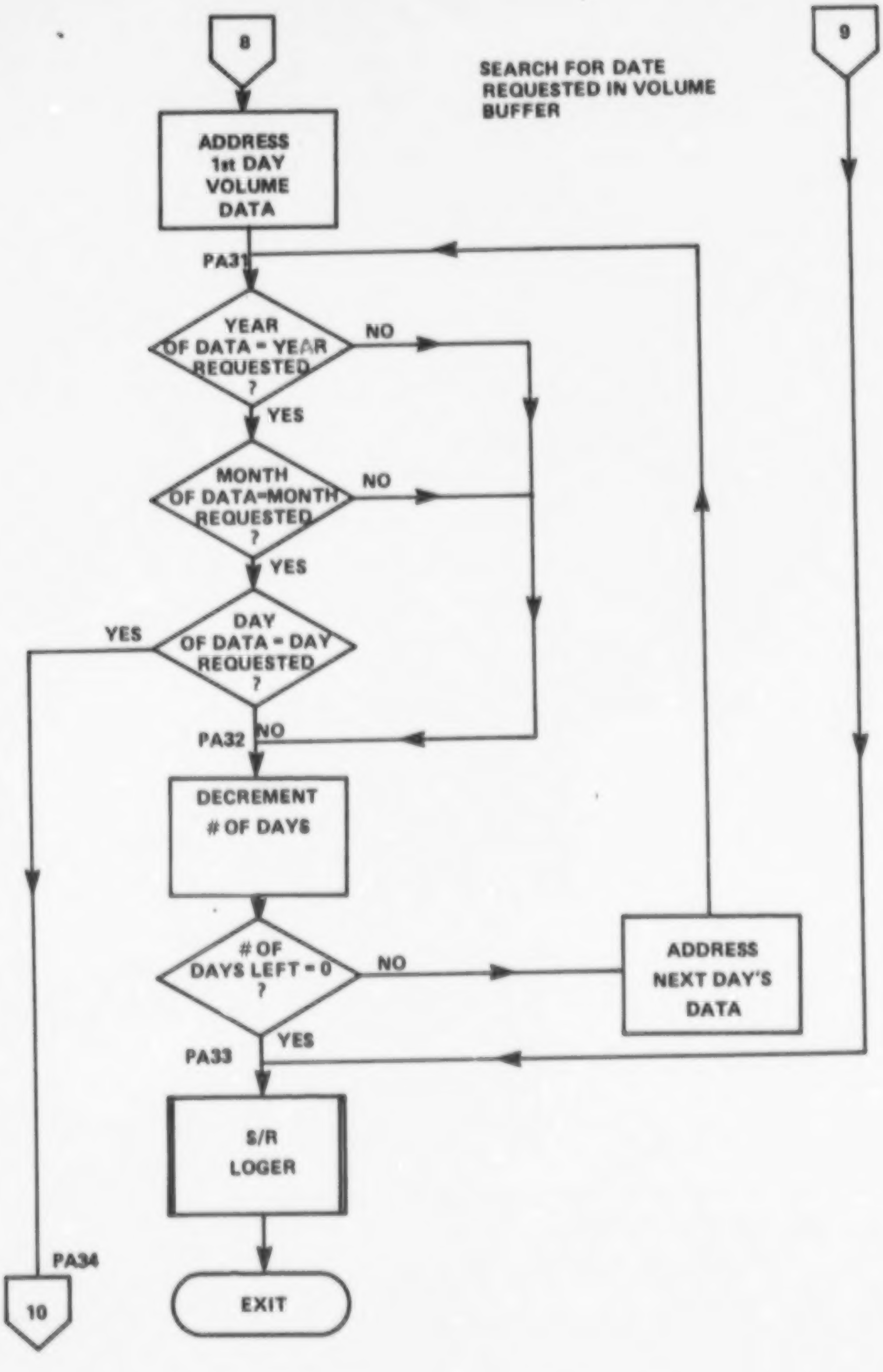
PREPARE ANSWER ROUTINE (PREANS)
SEND FIELD STATION MEMORY DATA, MESSAGE 001


2
CORRECT AMOUNT OF INPUT DATA ?
NO
YES
STARTING ADD. ≤ END OF RAM ?
NO
PA11
YES
STARTING ADD. ≥ BEGINNING OF PROM ?
NO
END ADD. ≤ END OF RAM ?
YES
END ADD. ≤ END OF PROM ?
NO
PA13
YES
SET UP RETURN MESSAGE
PA12
S/R LOGER
INITIALIZE BUFFERS FOR DATA TRANSMISSION
EXIT
EXIT

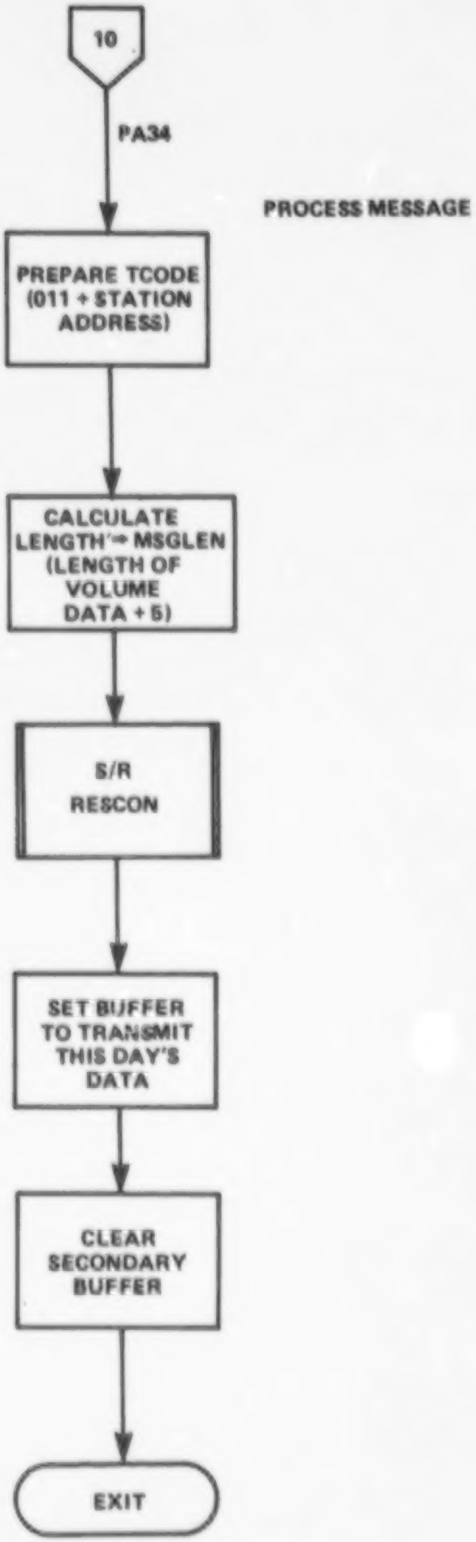
PREPARE ANSWER ROUTINE (PREANS)
SEND HOURLY VOLUME COUNT DATA (011)


3
CORRECT AMOUNT OF INPUT DATA ?
NO
YES
# OF DAYS VOLUME DATA AVAILABLE >0 ?
NO
YES
# OF DAYS VOLUME DATA < MAX FOR GIVEN RAM ?
NO
YES
PA33
8
9

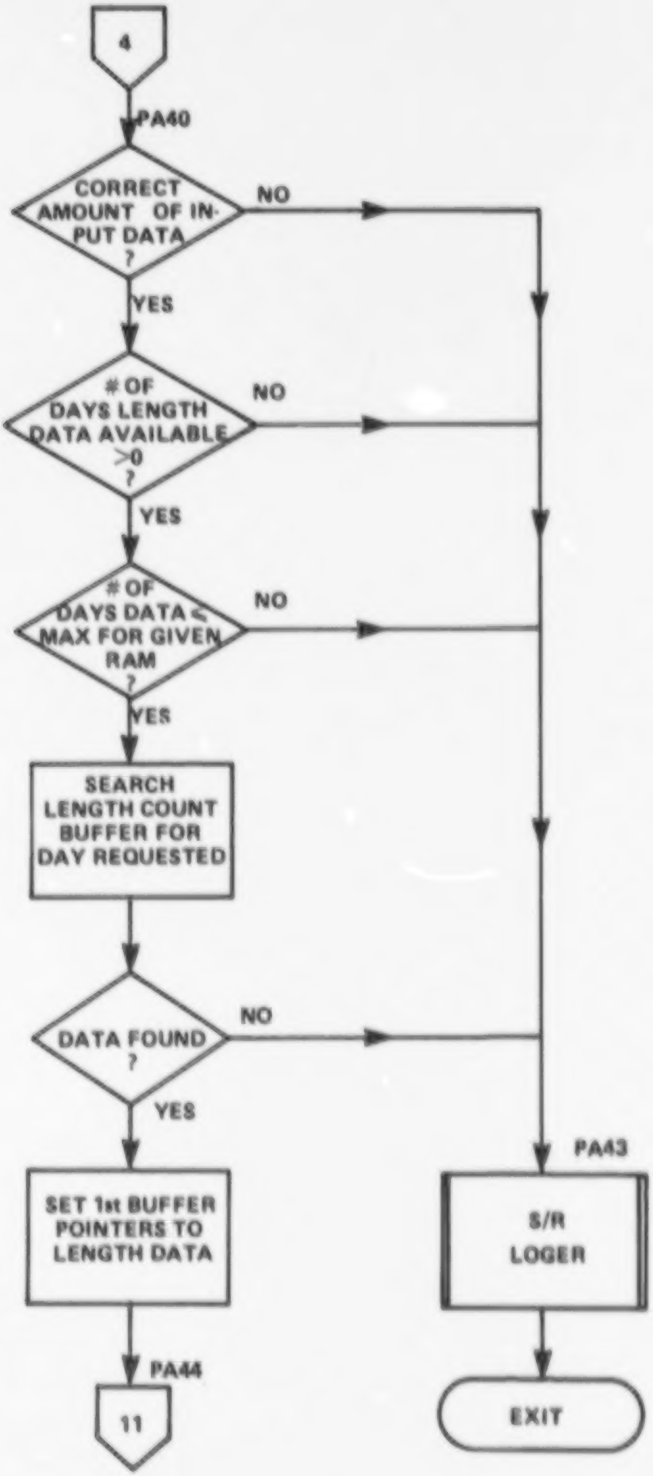
PREPARE ANSWER ROUTINE (PREANS)
SEND HOURLY VOLUME COUNT DATA (011)


8
9
SEARCH FOR DATE REQUESTED IN VOLUME BUFFER
ADDRESS 1st DAY VOLUME DATA
PA31
YEAR OF DATA = YEAR REQUESTED ?
NO
YES
MONTH OF DATA=MONTH REQUESTED ?
NO
YES
DAY OF DATA = DAY REQUESTED ?
YES
NO
PA32
DECREMENT # OF DAYS
# OF DAYS LEFT = 0 ?
NO
ADDRESS NEXT DAY'S DATA
YES
PA33
S/R LOGER
EXIT
PA34
10

SEND HOURLY VOLUME COUNT DATA (011)

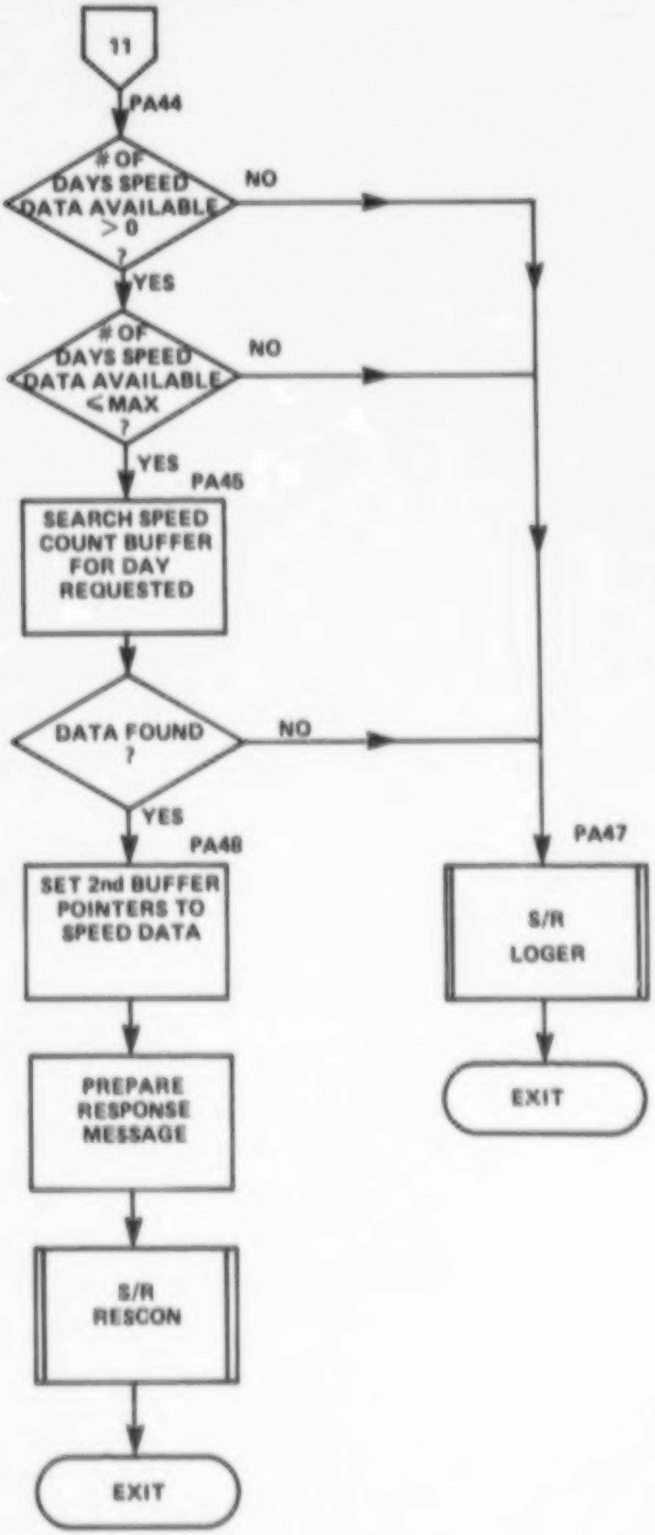

10
PA34
PROCESS MESSAGE
PREPARE TCODE (011 + STATION ADDRESS)
CALCULATE LENGTH ⇒ MSGLEN (LENGTH OF VOLUME DATA + 5)
S/R RESCON
SET BUFFER TO TRANSMIT THIS DAY'S DATA
CLEAR SECONDARY BUFFER
EXIT

PREPARE ANSWER ROUTINE (PREANS)
SEND LENGTH COUNT AND SPEED DATA (100)

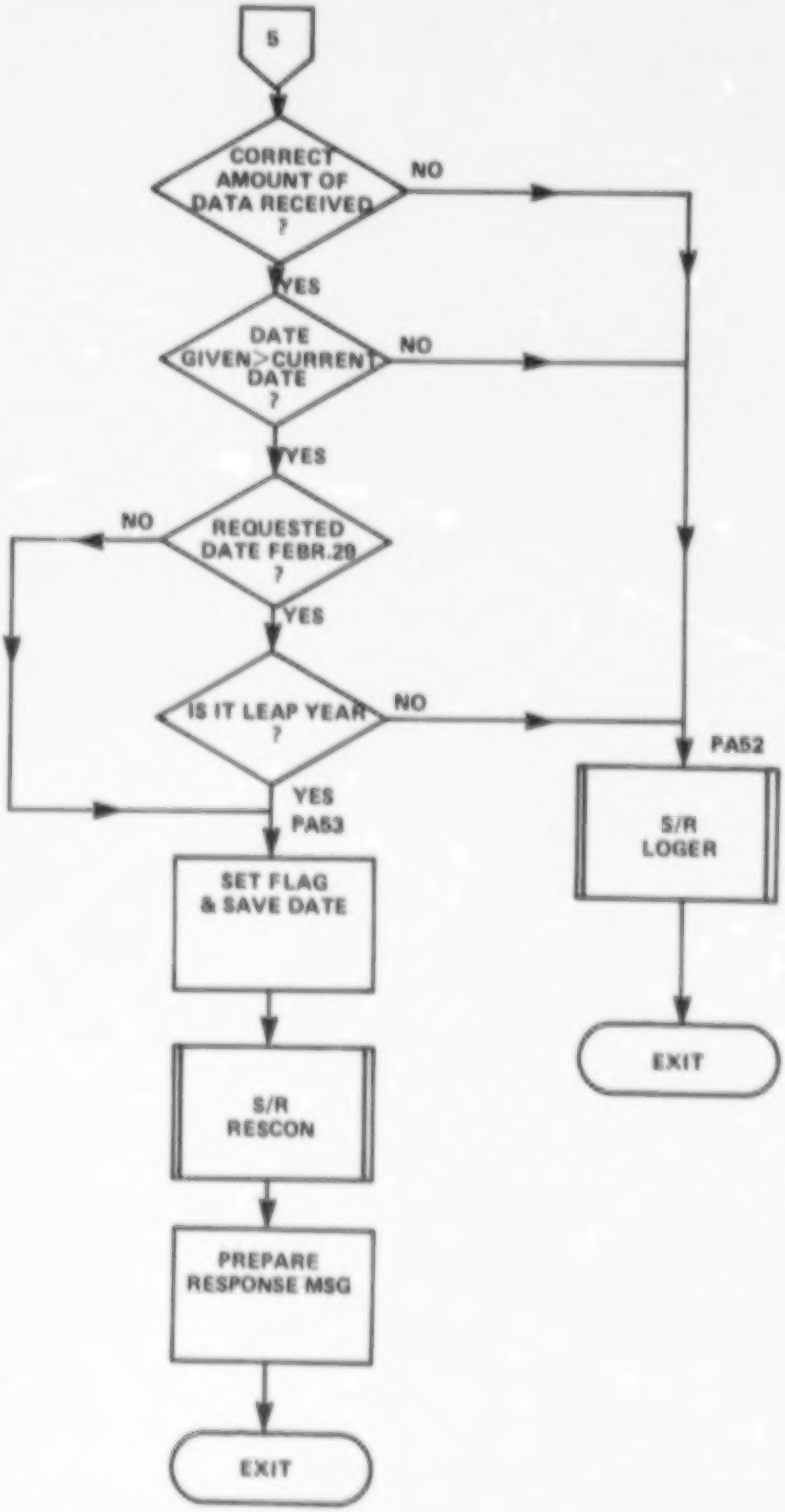

4
PA40
CORRECT AMOUNT OF INPUT DATA ?
NO
YES
# OF DAYS LENGTH DATA AVAILABLE >0 ?
NO
YES
# OF DAYS DATA ≤ MAX FOR GIVEN RAM ?
NO
YES
SEARCH LENGTH COUNT BUFFER FOR DAY REQUESTED
DATA FOUND ?
NO
YES
SET 1st BUFFER POINTERS TO LENGTH DATA
PA44
11
PA43
S/R LOGER
EXIT

PREPARE ANSWER ROUTINE (PREANS)
SEND LENGTH COUNT AND SPEED DATA (100)

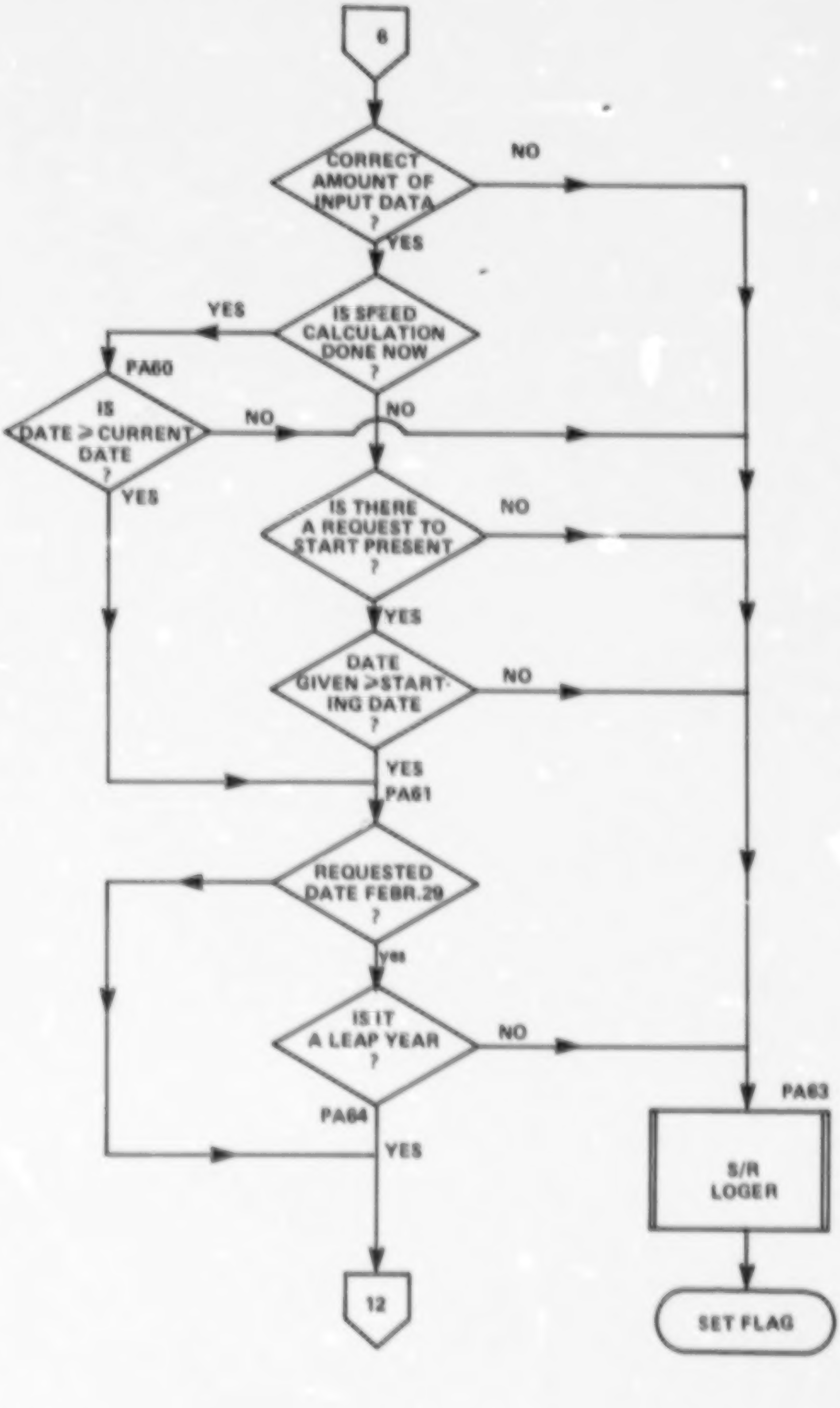
Page 8 of 15


11
PA44
# OF DAYS SPEED DATA AVAILABLE > 0 ?
NO
YES
# OF DAYS SPEED DATA AVAILABLE < MAX ?
NO
YES
PA45
SEARCH SPEED COUNT BUFFER FOR DAY REQUESTED
DATA FOUND ?
NO
YES
PA46
SET 2nd BUFFER POINTERS TO SPEED DATA
PREPARE RESPONSE MESSAGE
S/R RESCON
EXIT
PA47
S/R LOGER
EXIT

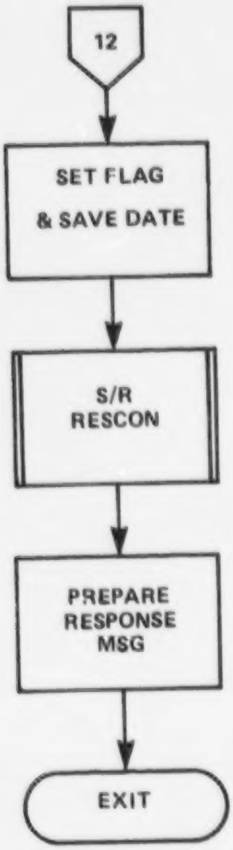
PREPARE ANSWER ROUTINE (PREANS)
START VEHICLE SPEED AND LENGTH DATA ACCUMULATION (101)


5
CORRECT AMOUNT OF DATA RECEIVED ?
NO
YES
DATE GIVEN>CURRENT DATE ?
NO
YES
NO
REQUESTED DATE FEBR.20 ?
YES
IS IT LEAP YEAR ?
NO
PA52
YES
PA53
S/R LOGER
SET FLAG & SAVE DATE
EXIT
S/R RESCON
PREPARE RESPONSE MSG
EXIT

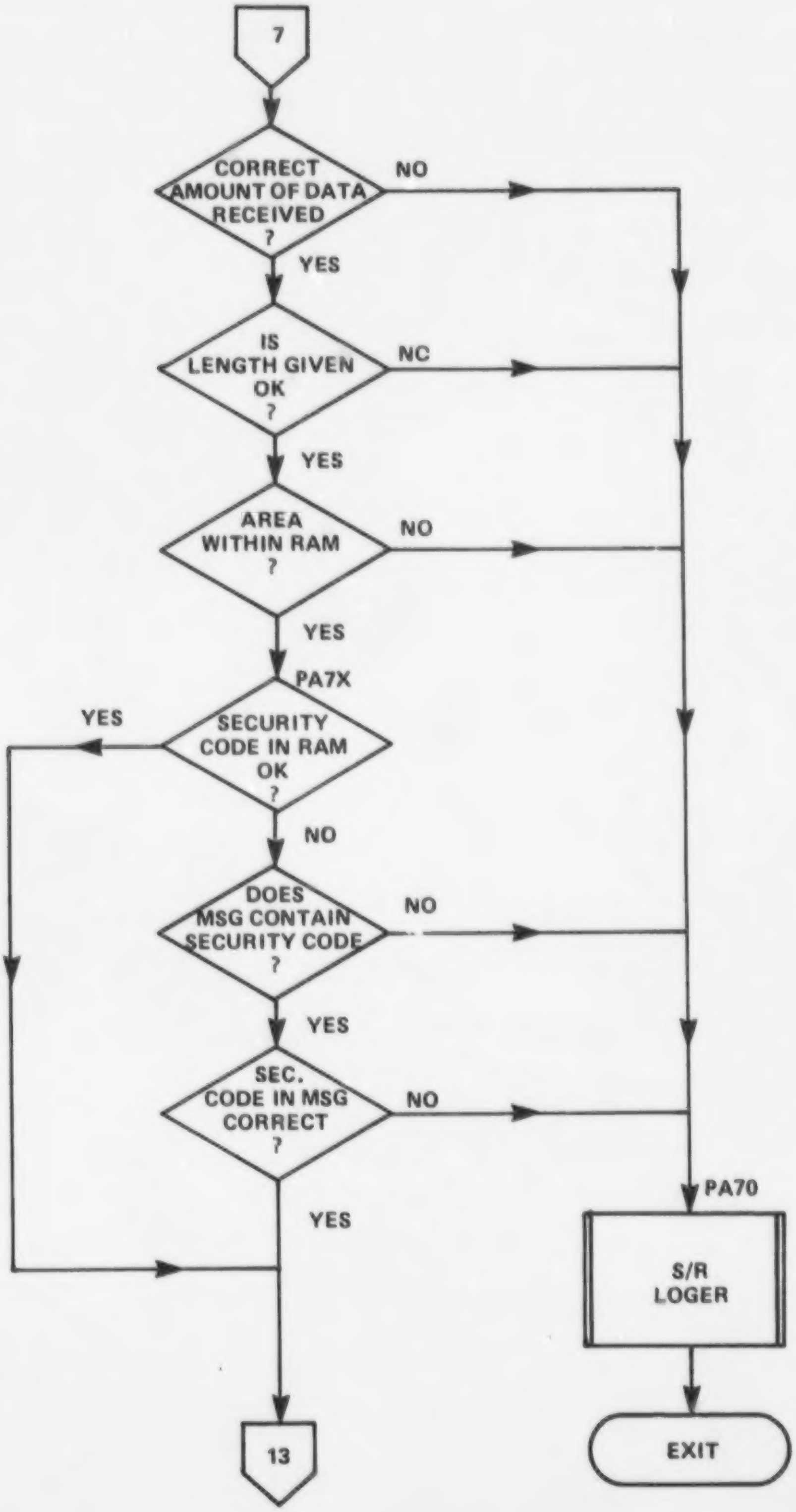
PREPARE ANSWER ROUTINE (PREANS)
END VEHICLE SPEED AND LENGTH DATA ACCUMULATION (110)


6
CORRECT AMOUNT OF INPUT DATA ?
NO
YES
IS SPEED CALCULATION DONE NOW ?
YES
PA60
IS DATE ≥ CURRENT DATE ?
NO
NO
YES
IS THERE A REQUEST TO START PRESENT ?
NO
YES
DATE GIVEN ≥ STARTING DATE ?
NO
YES
PA61
REQUESTED DATE FEBR.29 ?
yes
IS IT A LEAP YEAR ?
NO
PA64
YES
PA63
S/R LOGER
12
SET FLAG

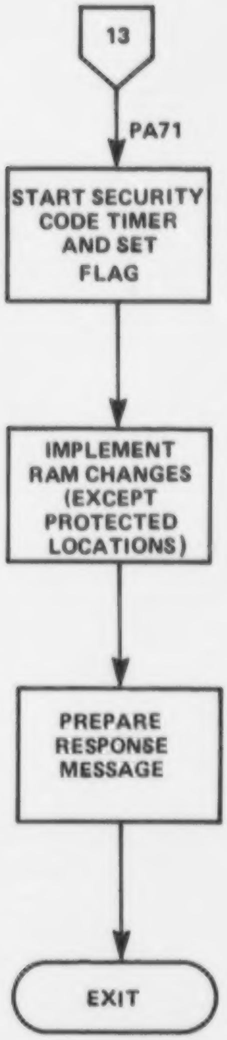
PREPARE ANSWER ROUTINE (PREANS)
END VEHICLE SPEED AND LENGTH DATA ACCUMULATION (110)


12
SET FLAG
& SAVE DATE
S/R
RESCON
PREPARE
RESPONSE
MSG
EXIT

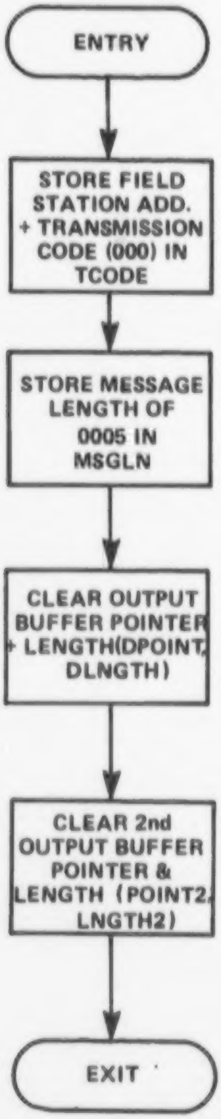
PREPARE ANSWER ROUTINE (PREANS)
CHANGE FIELD STATION MEMORY (111)


7
CORRECT AMOUNT OF DATA RECEIVED ?
NO
YES
IS LENGTH GIVEN OK ?
NC
YES
AREA WITHIN RAM ?
NO
YES
PA7X
SECURITY CODE IN RAM OK ?
YES
NO
DOES MSG CONTAIN SECURITY CODE ?
NO
YES
SEC. CODE IN MSG CORRECT ?
NO
YES
PA70
S/R LOGER
EXIT
13

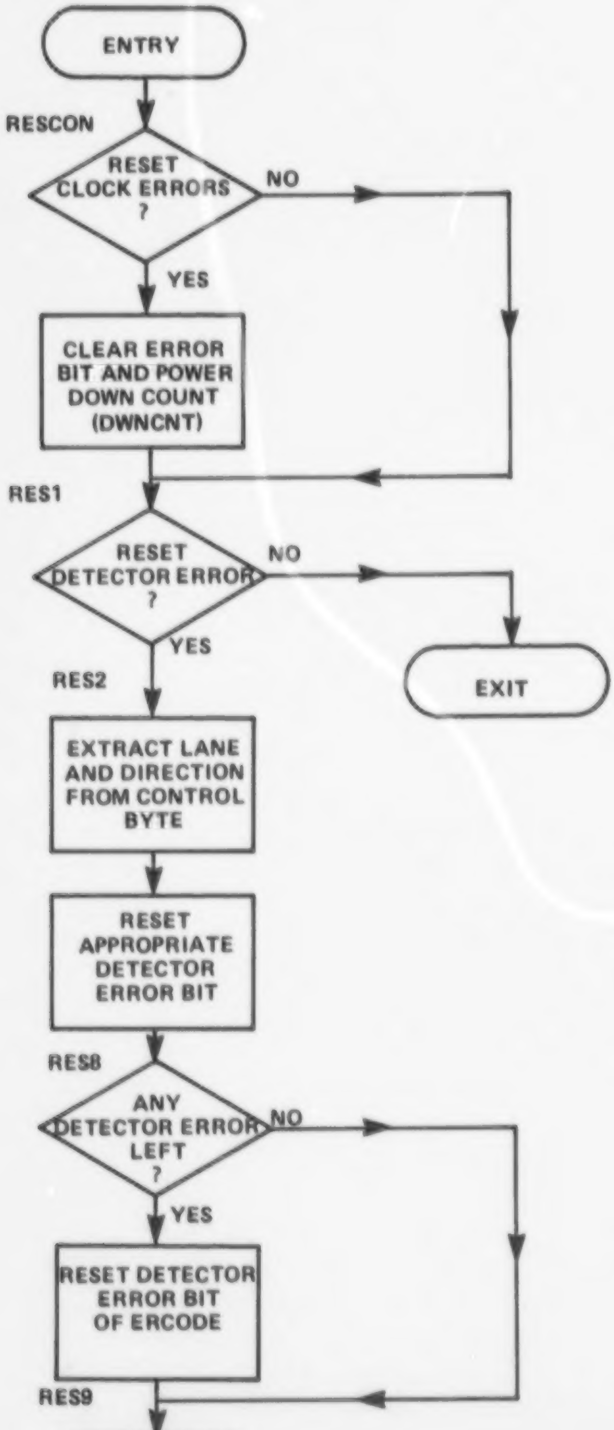
PREPARE ANSWER ROUTINE (PREANS)
CHANGE FIELD STATION MEMORY (111)


13
PA71
START SECURITY CODE TIMER AND SET FLAG
IMPLEMENT RAM CHANGES (EXCEPT PROTECTED LOCATIONS)
PREPARE RESPONSE MESSAGE
EXIT

PREPARE ANSWER ROUTINE (PREANS)
LOGICAL ERROR MESSAGE SUBROUTINE (LOGER)


ENTRY
STORE FIELD STATION ADD. + TRANSMISSION CODE (000) IN TCODE
STORE MESSAGE LENGTH OF 0005 IN MSGLN
CLEAR OUTPUT BUFFER POINTER + LENGTH(DPOINT, DLNGTH)
CLEAR 2nd OUTPUT BUFFER POINTER & LENGTH (POINT2, LNGTH2)
EXIT

PREPARE ANSWER ROUTINE (PREANS)
RESET CONTROL SUBROUTINE (RESCON)


ENTRY
RESCON
RESET CLOCK ERRORS ?
NO
YES
CLEAR ERROR BIT AND POWER DOWN COUNT (DWNCNT)
RES1
RESET DETECTOR ERROR ?
NO
YES
EXIT
RES2
EXTRACT LANE AND DIRECTION FROM CONTROL BYTE
RESET APPROPRIATE DETECTOR ERROR BIT
RES8
ANY DETECTOR ERROR LEFT ?
NO
YES
RESET DETECTOR ERROR BIT OF ERCODE
RES9
EXIT

7.6.6/ Communications Timer Routine (TIMER1)

Purpose

This routine resolves deadlock situations resulting from communication time-outs. It is executed every 1/10 sec.

The routine uses the 1/10 sec counters CTIME1 and CTIME2. CTIME1 is used to resolve the receiving deadlock that occurs if there is an excessively long delay (3 seconds) during receipt of a message. CTIME2 is used to resolve the transmitting deadlock that occurs if there is a delay (exceeding 3/10 sec) due to the loss of the data carrier. Note that a data carrier loss is undetectable while transmitting.

The routine maintains the two timers and determines if there is a deadlock situation (i.e., one of the timers has expired). It then takes appropriate action. In the case of the receive time-out, TIMER1 initiates transmission of a reply to the central computer. If the transmit timer times out, the controller enters a receiving state.

Description

The communication system can be in one of three different states:

1/ RBITS = 1 -- waiting for a message after power-up, after a previous message or after a data carrier loss during transmission.

2/ RBITS greater than 1 -- receiving a message.

3/ RBITS = 0 -- transmitting a message.

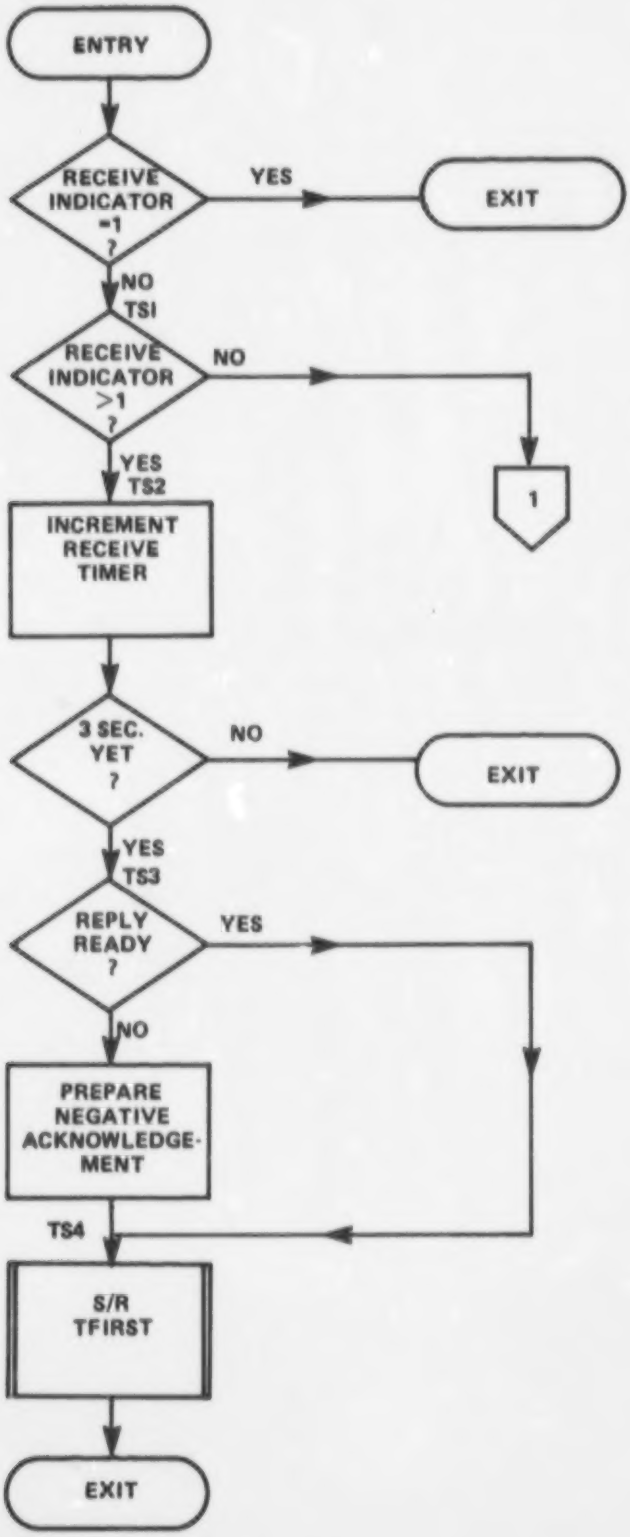
The state is determined by examining RBITS.

If the controller is in state 1, the routine TIMER1 does nothing, it simply terminates (since no communication is currently taking place).

If the controller is in state 2, the receive timer CTIME1 is incremented. If CTIME1 has not yet reached 3 seconds, the routine terminates. Otherwise, we have a receive time-out. If a reply message has not already been prepared (indicated by the value of RBITS), then TIMER1 prepares a negative acknowledgement message. In any case, the transmission of the message is initiated by calling TFIRST and the routine then terminates.

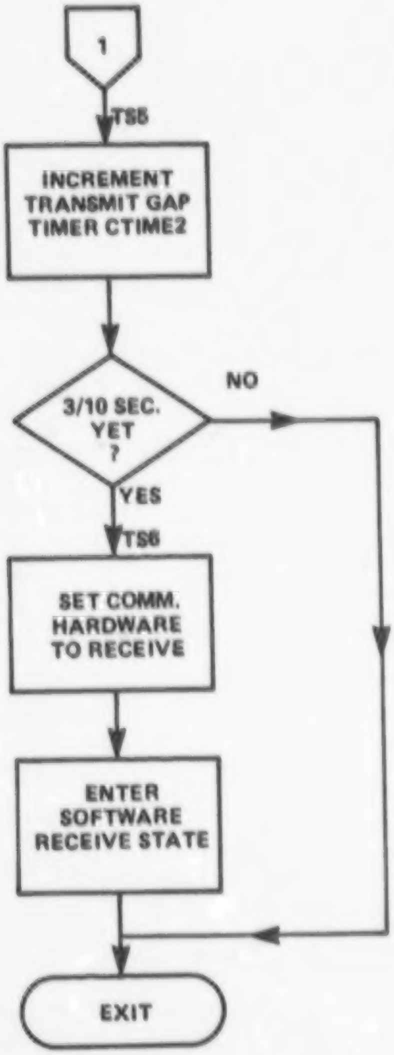
COMMUNICATION TIMER ROUTINE (TIMER 1)

Page 1 of 2


ENTRY
RECEIVE INDICATOR =1 ?
YES
EXIT
NO
TSI
RECEIVE INDICATOR >1 ?
NO
1
YES
TS2
INCREMENT RECEIVE TIMER
3 SEC. YET ?
NO
EXIT
YES
TS3
REPLY READY ?
YES
NO
PREPARE NEGATIVE ACKNOWLEDGE-MENT
TS4
S/R TFIRST
EXIT

COMMUNICATION TIMER ROUTINE (TIMER 1)

Page 2 of 2


1
TS5
INCREMENT TRANSMIT GAP TIMER CTIME2
3/10 SEC. YET ?
NO
YES
TS6
SET COMM. HARDWARE TO RECEIVE
ENTER SOFTWARE RECEIVE STATE
EXIT

If the controller is in state 3, the transmit timer CTIME2 is incremented. If CTIME2 has not yet reached 3/10 sec the routine terminates. Otherwise, we have a transmit time-out, and TIMER1 puts the controller into a receive state.

7.6.7/ Security Code Timer Routine (TIMER2)

Purpose
This routine resets the security code 255 minutes after a message from the central computer has changed RAM memory. It is executed every minute and updates the security code timer.

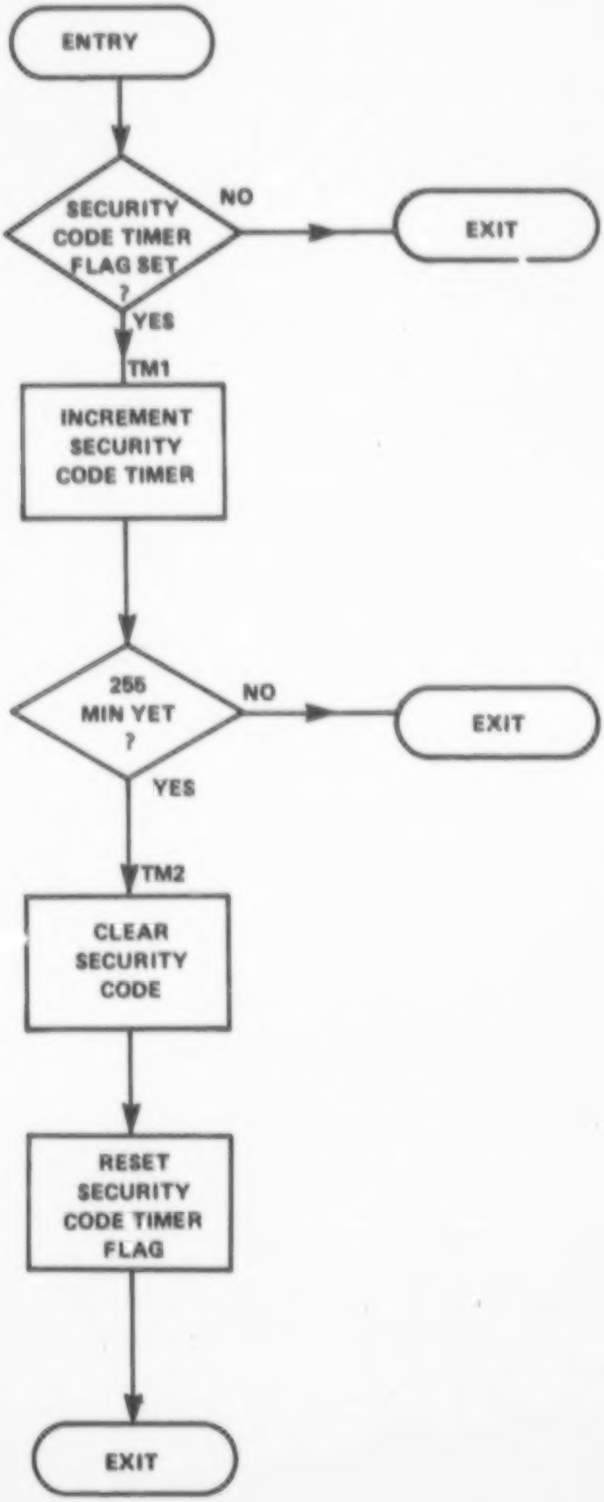
Description
If the security code timer flag (bit 4 of MFLAG) is not set, the routine terminates. Note that this flag is set by routine PREANS in response to a "change field station memory" request.

Otherwise, the security code timer CTIME3 is incremented. If it has not yet reached 255 min, the routine terminates.

If CTIME3 has reached 255 min, the security code is cleared, preventing unauthorized RAM changes until the code is set again. Also, the security code timer flag is reset. The routine then terminates.

SECURITY CODE TIMER ROUTINE (TIMER2)

Page 1 of 1


ENTRY
SECURITY CODE TIMER FLAG SET ?
NO
EXIT
YES
TM1
INCREMENT SECURITY CODE TIMER
256 MIN YET ?
NO
EXIT
YES
TM2
CLEAR SECURITY CODE
RESET SECURITY CODE TIMER FLAG
EXIT

7.7/ Master Initialization Routine (MASINT)

Purpose

This routine initializes system and application variables as required. It is started every second and executed if the master flag is set. This flag should be set (via the keyboard or a communication message) after the station number, time, date and station geometry have been initialized.

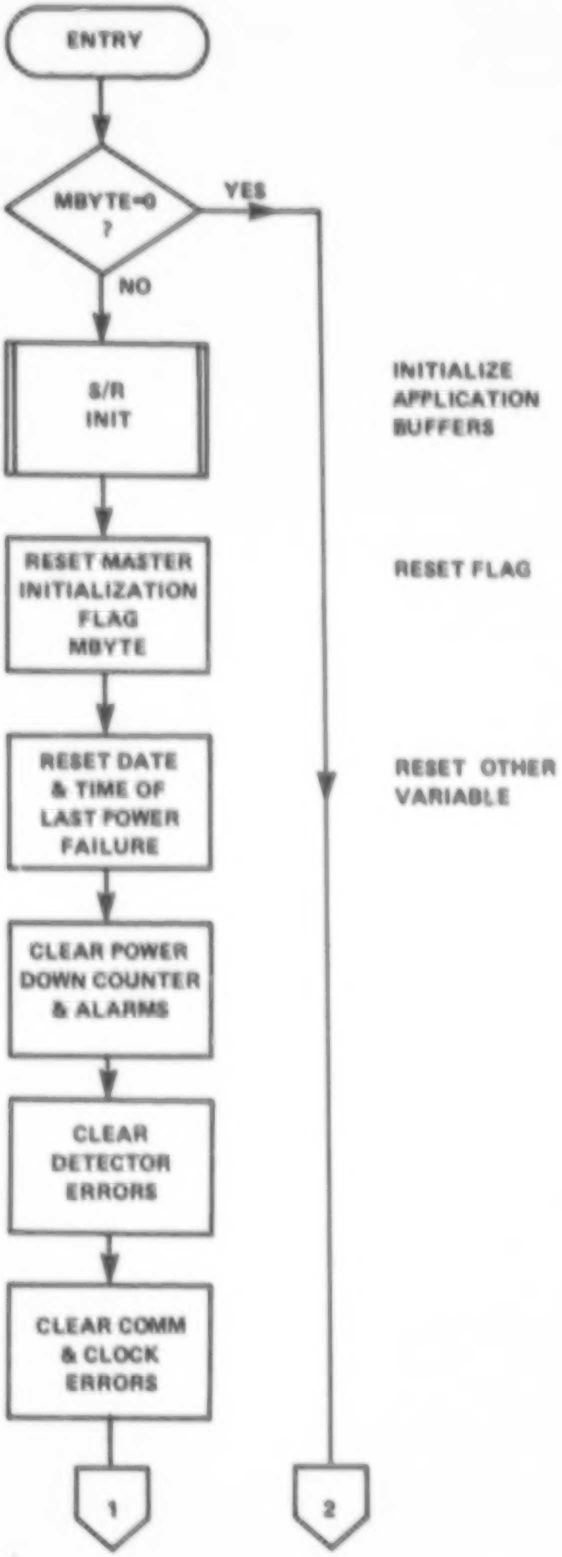
Description

The master initialization flag is examined. If it is zero, the routine terminates.

Otherwise, the routine INIT is called to initialize application variables and buffers. The master initialization flag is then reset to indicate the initialization has been done. The date and time of the last power-down failure are cleared, and the power-down counter is also cleared. Then, the power-down alarms are reset.

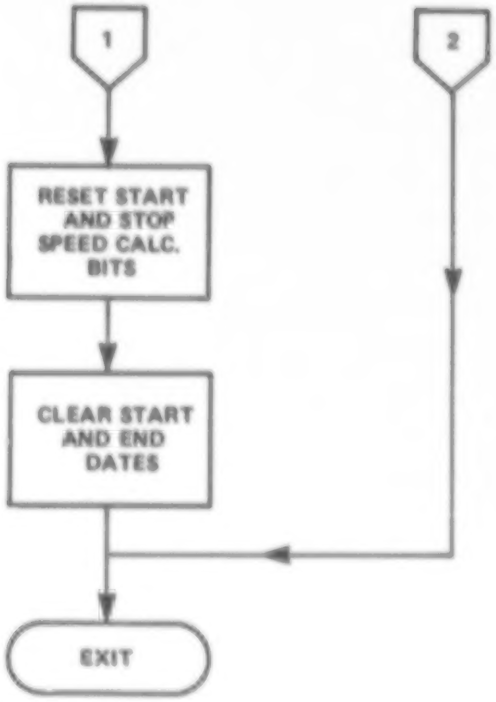
The four detector error words and the communication error word are all cleared. Finally, the clock error bit of ERCODE is reset. In addition, the start and stop speed calculation bits of MFLAG are reset and the start and end dates are cleared. The routine then terminates.

MASTER INITIALIZATION ROUTINE (MASINT) Page 1 of 2


ENTRY
MBYTE=0 ?
YES
NO
S/R INIT
INITIALIZE APPLICATION BUFFERS
RESET MASTER INITIALIZATION FLAG MBYTE
RESET FLAG
RESET DATE & TIME OF LAST POWER FAILURE
RESET OTHER VARIABLE
CLEAR POWER DOWN COUNTER & ALARMS
CLEAR DETECTOR ERRORS
CLEAR COMM & CLOCK ERRORS
1
2

MASTER INITIALIZATION ROUTINE (MASINT)

Page 2 of 2


1
2
RESET START AND STOP SPEED CALC. BITS
CLEAR START AND END DATES
EXIT

RESET "START" AND "STOP" SPEED CALCULATION VARIABLES

7.8/ Initiate PROM Checksum Routine (CHECK24)

Purpose

This routine initiates a PROM checksum calculation to ensure the integrity of PROM. It is executed every 24 hours at midnight. The checksum then proceeds at 1 byte every 1/60 sec until it is complete.

Description

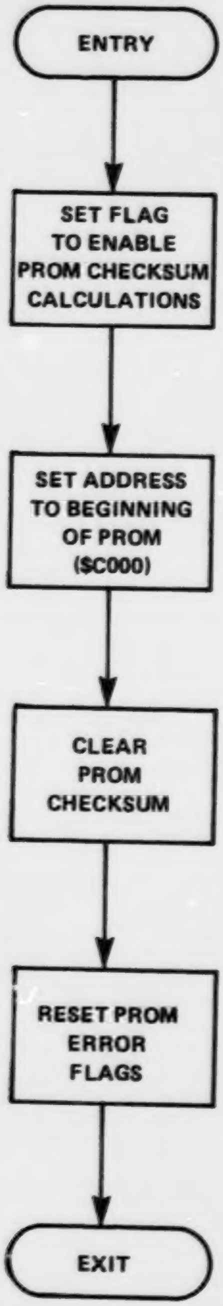
CHECK24 sets bit 5 of MFLAG to indicate a checksum should be performed. This indicates to CHEK60 that the checksum can proceed. It also indicates to CALPRO that the checksum is to be computed at 1 byte every 1/60 sec instead of doing all of PROM at once.

The starting PROM address is set to $C000.

The checksum itself is initialized to 0.

Finally, the error flag byte PRERFL, which indicates those chips with an invalid checksum (i.e., unexpected checksum) is cleared.

INITIATE PROM CHECKSUM ROUTINE (CHEK24) Page 1 of 1


ENTRY
SET FLAG TO ENABLE PROM CHECKSUM CALCULATIONS
SET ADDRESS TO BEGINNING OF PROM ($C000)
CLEAR PROM CHECKSUM
RESET PROM ERROR FLAGS
EXIT

7.8.1/ Perform Checksum 1 Byte Subroutine (CHECK60)

Purpose

This routine, executed every 1/60 sec, simply checks to see if a PROM checksum has been completed for this day. If it has not, CALPRO is invoked to add one byte of PROM memory to the checksum.

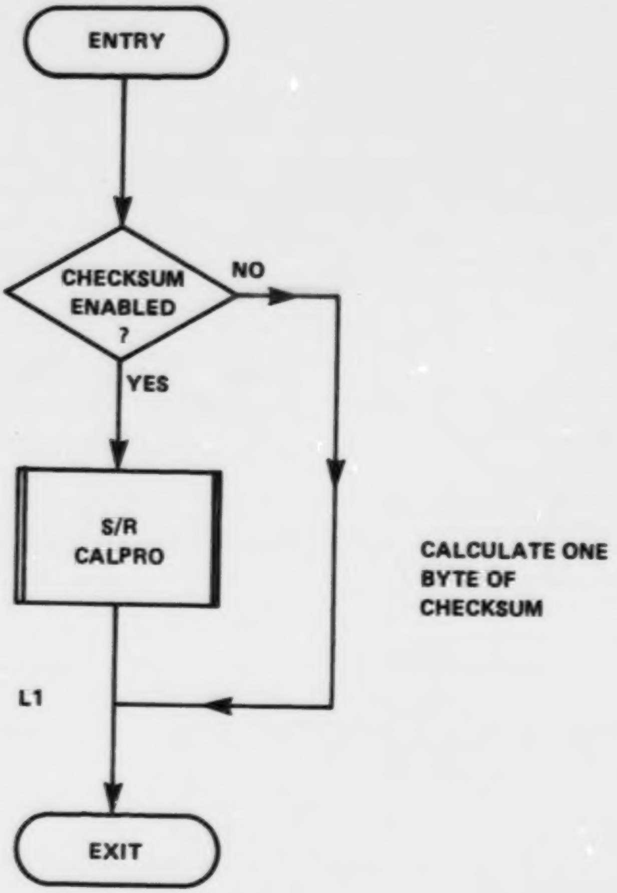
Description

CHEK60 examines bit 5 of MFLAG. If it is set, then the checksum is updated by invoking the routine CALPRO. If the bit is not set, a checksum has already been completed today so no further processing is necessary.

Note that the PROM checksum flag in MFLAG is set by CHECK24 every 24 hours and reset by CALPRO at the end of the PROM checksum calculations. It is also set by POWRON, but POWRON calls CALPRO directly, so CHECK60 is not involved in the power-up PROM checksum verification.

UPDATE PROM CHECKSUM BY ONE BYTE SUBROUTINE (CHEK60)

Page 1 of 1


ENTRY
CHECKSUM ENABLED ?
NO
YES
S/R CALPRO
CALCULATE ONE BYTE OF CHECKSUM
L1
EXIT

7.8.2/ Compute PROM Checksum Subroutine (CALPRO)

Purpose

This routine calculates the checksum for a 2K PROM chip. It does this by taking one byte of PROM and adding it into a PROM checksum. At the end of every 2K of PROM, the checksum is compared to the expected checksum (the way the expected checksum is calculated is described in Appendix B). If they are not the same, an error is indicated for that chip. This means that somehow the PROM contents has changed (due to deterioration or damage of the chip).

On power-up, CALPRO is called from POWRON and the above tests are performed for all of PROM. Also, every 24 hours the calculations are started by CHEK24. CALPRO is then called by CHEK60 every 1/60 sec until the checksum is complete, each call only contributing one byte to the checksum.

When the PROM checksum calculations have been completed, a RAM test is initiated.

Description

Using the pointer PADDR, a byte is fetched from PROM memory and is added to PCHECK, a 16-bit quantity. Thus, the checksum, when finished, is the least 16 significant bits of the sum of all data bytes in one 2K PROM chip. There are, at most, 8 such chips.

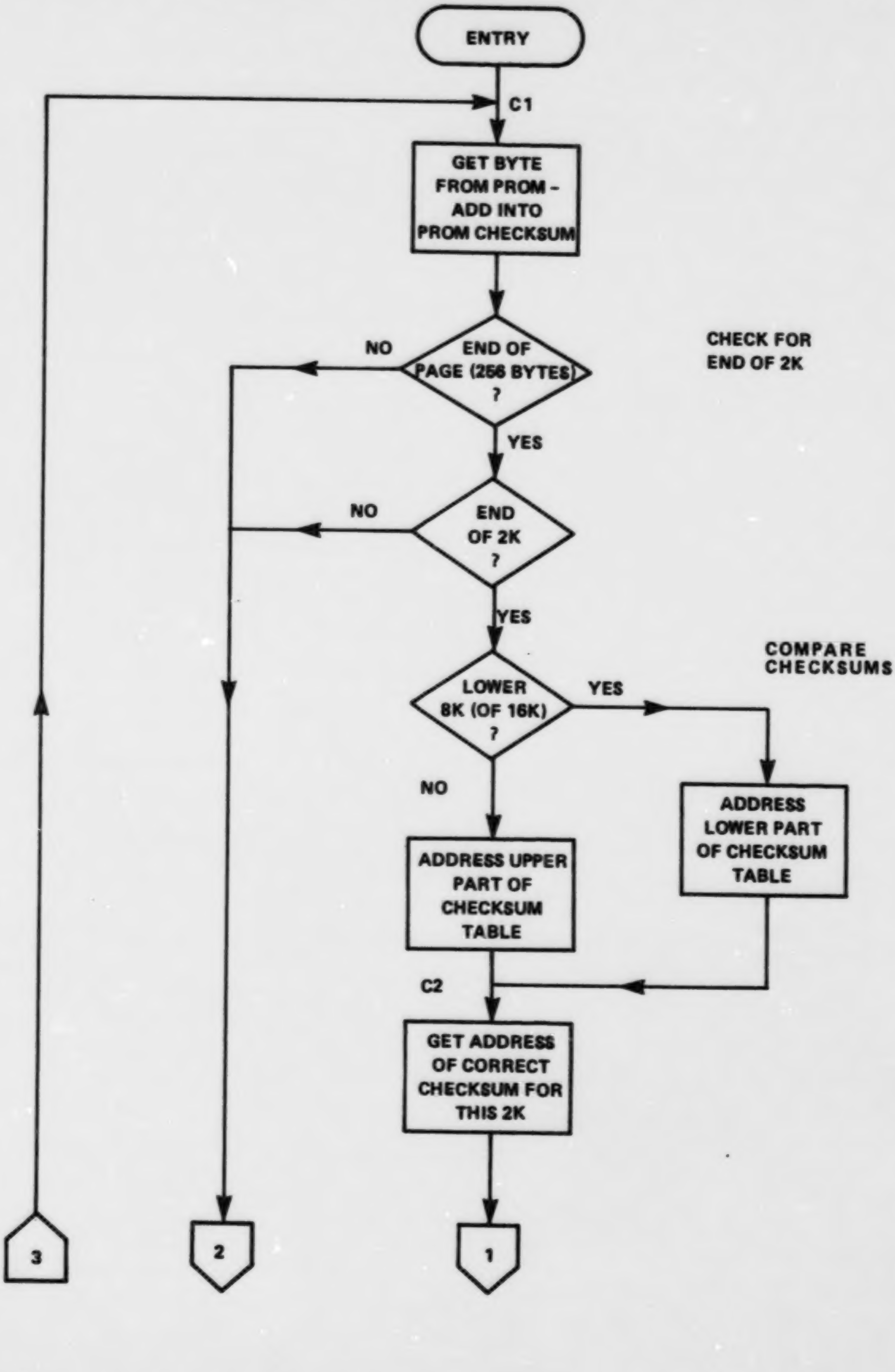
At the end of 2K (indicated by $7FF or $FFF in the least significant three nibbles of the address), the checksum PCHECK is compared to the expected value obtained during PROM programming. The correct values are stored in the table PCHKSM. (Note that empty 2K sockets have a checksum of $F800). If the computed checksum differs from the expected checksum, a bit is set in PRERFL to indicate that this chip is faulty.

If all of PROM ($C000-$FFFF) has been checked, then the results of the test are saved in ERCODE. The PROM error bit is set if there are any errors and reset otherwise. In either case, bit 5 of MFLAG is reset to prevent further PROM checks for today, bit 6 of MFLAG is set to initiate a RAM test, and the RAM test variables are initialized.

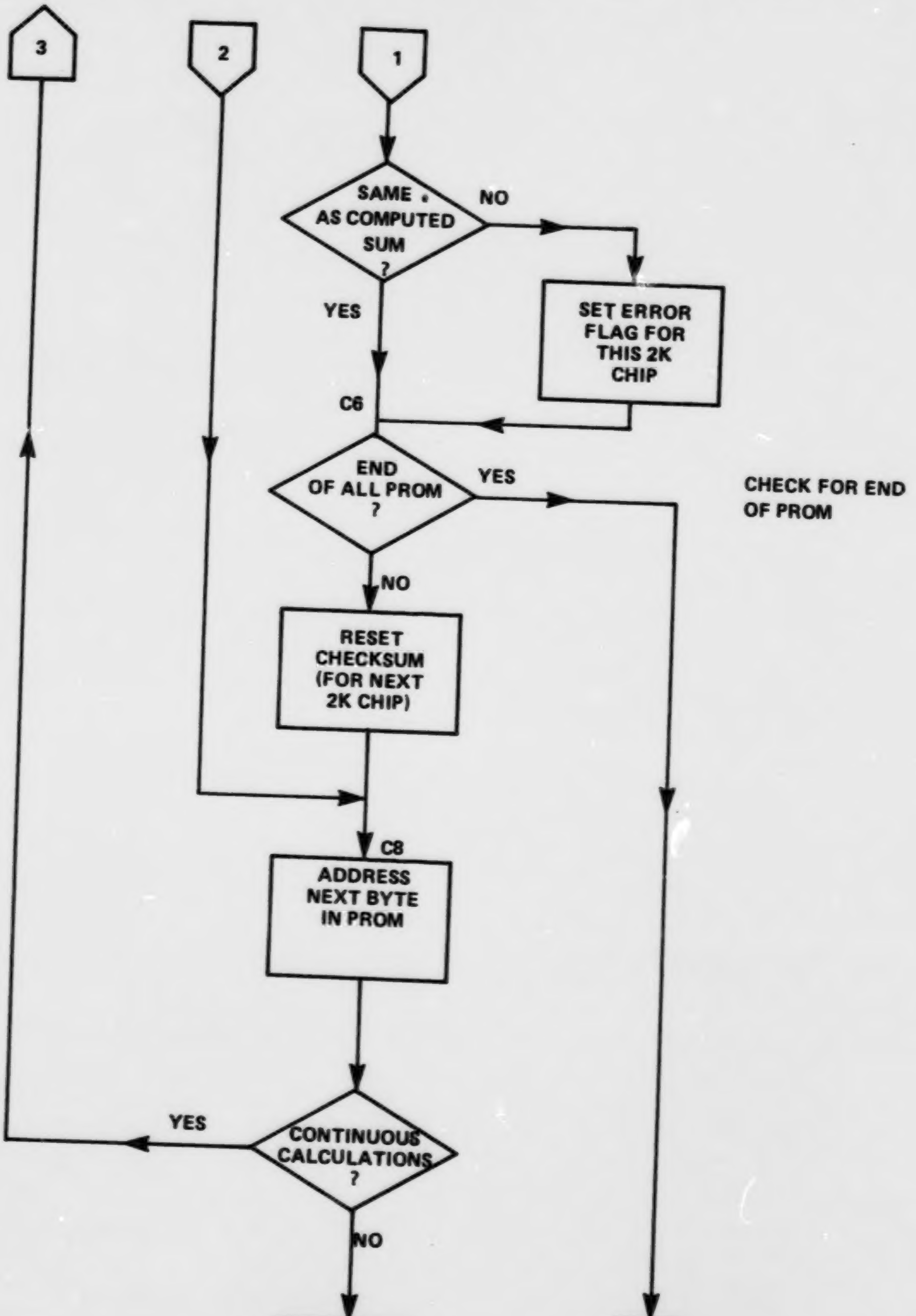
If there is still PROM to be checked, PADDR is incremented to address the next PROM byte. Then, if bit 5 of MFLAG is not set, this is a power-up PROM check so CALPRO loops back to the beginning and repeats the checksum calculations. Otherwise, CALPRO terminates.

CALCULATE PROM CHECKSUM SUBROUTINE (CALPRO)

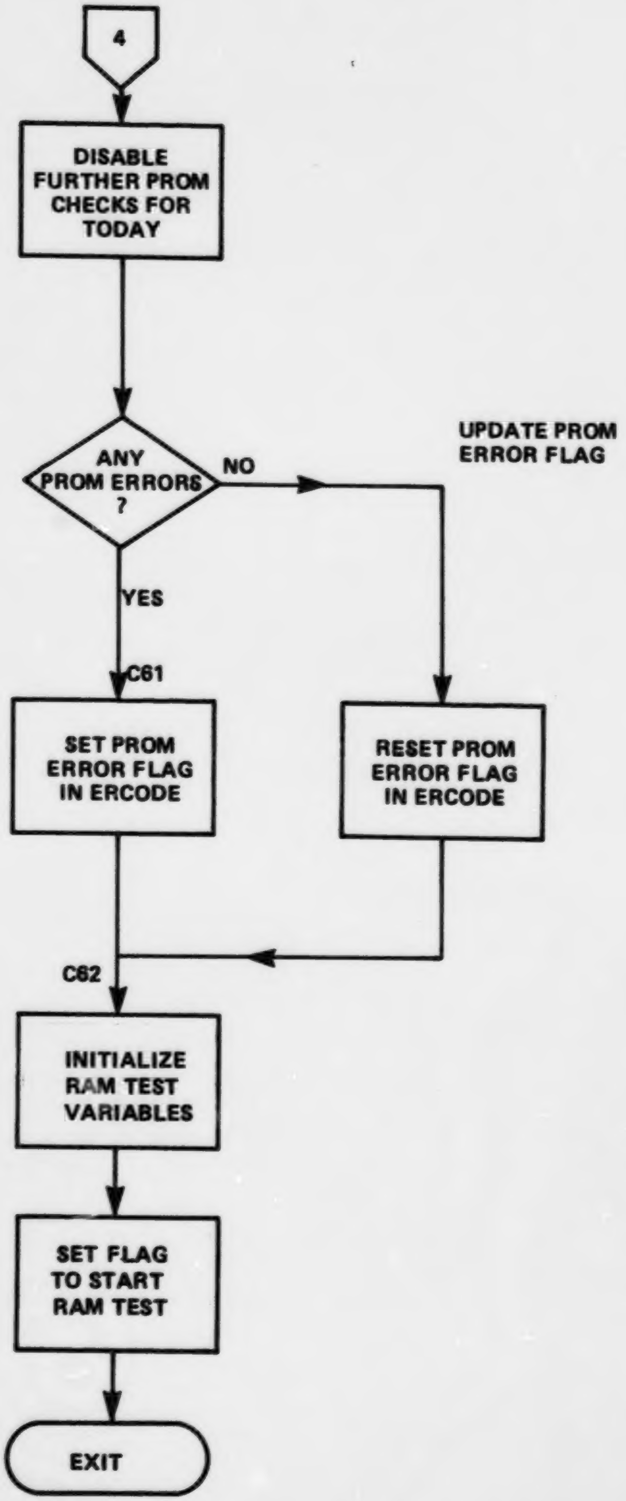
Page 1 of 3


ENTRY
C1
GET BYTE FROM PROM - ADD INTO PROM CHECKSUM
NO
END OF PAGE (256 BYTES) ?
CHECK FOR END OF 2K
YES
NO
END OF 2K ?
YES
COMPARE CHECKSUMS
LOWER 8K (OF 16K) ?
YES
NO
ADDRESS LOWER PART OF CHECKSUM TABLE
ADDRESS UPPER PART OF CHECKSUM TABLE
C2
GET ADDRESS OF CORRECT CHECKSUM FOR THIS 2K
3
2
1

CALCULATE PROM CHECKSUM SUBROUTINE (CALPRO) Page 2 of 3


3
2
1
SAME AS COMPUTED SUM ?
NO
SET ERROR FLAG FOR THIS 2K CHIP
YES
C6
END OF ALL PROM ?
YES
CHECK FOR END OF PROM
NO
RESET CHECKSUM (FOR NEXT 2K CHIP)
C8
ADDRESS NEXT BYTE IN PROM
YES
CONTINUOUS CALCULATIONS ?
NO
EXIT
4

CALCULATE PROM CHECKSUM SUBROUTINE (CALPRO)


4
DISABLE FURTHER PROM CHECKS FOR TODAY
ANY PROM ERRORS ?
NO
UPDATE PROM ERROR FLAG
YES
C61
SET PROM ERROR FLAG IN ERCODE
RESET PROM ERROR FLAG IN ERCODE
C62
INITIALIZE RAM TEST VARIABLES
SET FLAG TO START RAM TEST
EXIT

7.9/ RAM Test Routines

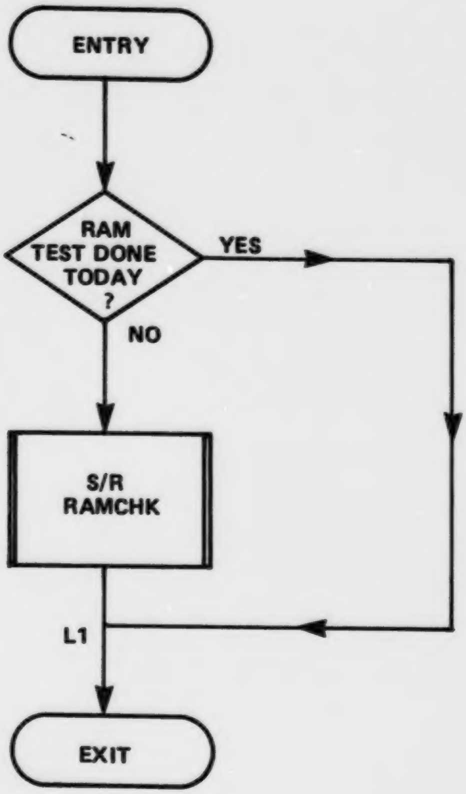
7.9.1/ Test 1 Byte of RAM Routine (RAM60)

Purpose

This routine is invoked every 1/60 sec. It simply tests to see if a RAM test has already been completed today. If it has, the routine terminates. If no test has been done today, routine RAMCHK is invoked to test one byte of RAM.

Description

Bit 6 of the master flag MFLAG is examined. If it is not set, a RAM check has already been performed today (or a PROM test is still in progress) so the routine terminates. Otherwise, subroutine RAMCHK is invoked to test one byte of RAM. Note that at power-up, the RAM test is performed differently in that RAMCHK is called directly by POWRON and the entire RAM check is done at once. The daily RAM check, started every 24 hours by CALPRO after it has finished the PROM test, proceeds at one byte every 1/60 sec until all of RAM has been tested.


ENTRY
RAM TEST DONE TODAY ?
YES
NO
S/R RAMCHK
L1
EXIT

7.9.2/ Perform RAM Test Routine (RAMCHK)

Purpose

This routine tests one byte of RAM when called by RAM60, or tests all of RAM when called by POWRON at power-up time. Each bit of the RAM locations is tested to make sure data are being properly saved. If any errors are detected, an error bit is set for a 2K RAM chip (there are ten such bits for, at most, ten 2K chips). Finally, at the end of the RAM test, the results (successful or unsuccessful) are summarized in ERCODE.

Description

Since a test value will be written to a RAM location, interrupts are first disabled. This way, variables cannot be altered by interrupts and, more importantly, if we are testing RAM locations associated with the system stack, the entire controller will not be hung.

Once interrupts have been disabled, the pointer PADDR is used to fetch one byte of RAM, and this byte is saved to restore the original data later. Then, every bit of the RAM location is exercised by writing 00000000, 00000001, 00000010, ..., 10000000 and 11111111 to the location and then reading back the value to make sure it was saved properly. The original data are then restored and interrupts are enabled.

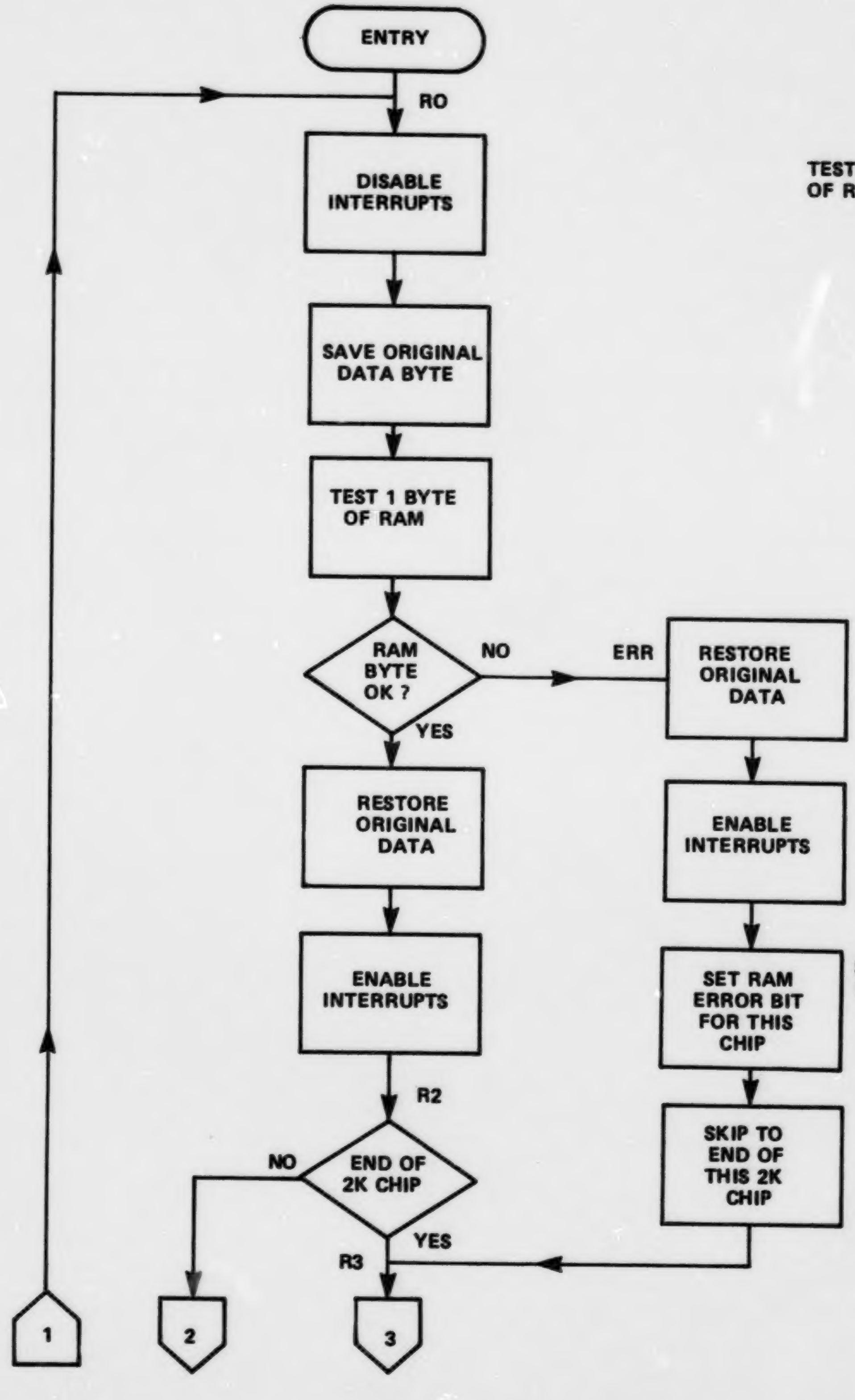
If an error occurred when comparing written and read values, an error bit is set in RAMERR (or RAMERR+1) and PADDR is set to the end of this 2K chip, since if there is one error in the chip there is no need to check the chip further.

If it is the end of a 2K chip, the test advances to the next chip. If, in fact, the entire RAM has been checked, the RAM test bit of MFLAG is reset to indicate today's RAM test has been completed, the RAM error indicator is set in ERCODE if there are any RAM errors (it is reset otherwise), and the routine terminates.

If all of RAM has not been tested, PADDR is incremented to address the next byte of RAM. Then, bit 6 of MFLAG is examined. If it is set, this RAM test was started after midnight by CALPRO at the end of the PROM test, so the routine terminates. If it is not set, this is a power-up RAM test and the routine loops back to the beginning and repeats the tests for the next byte.

PERFORM RAM TEST ROUTINE (RAMCHK)

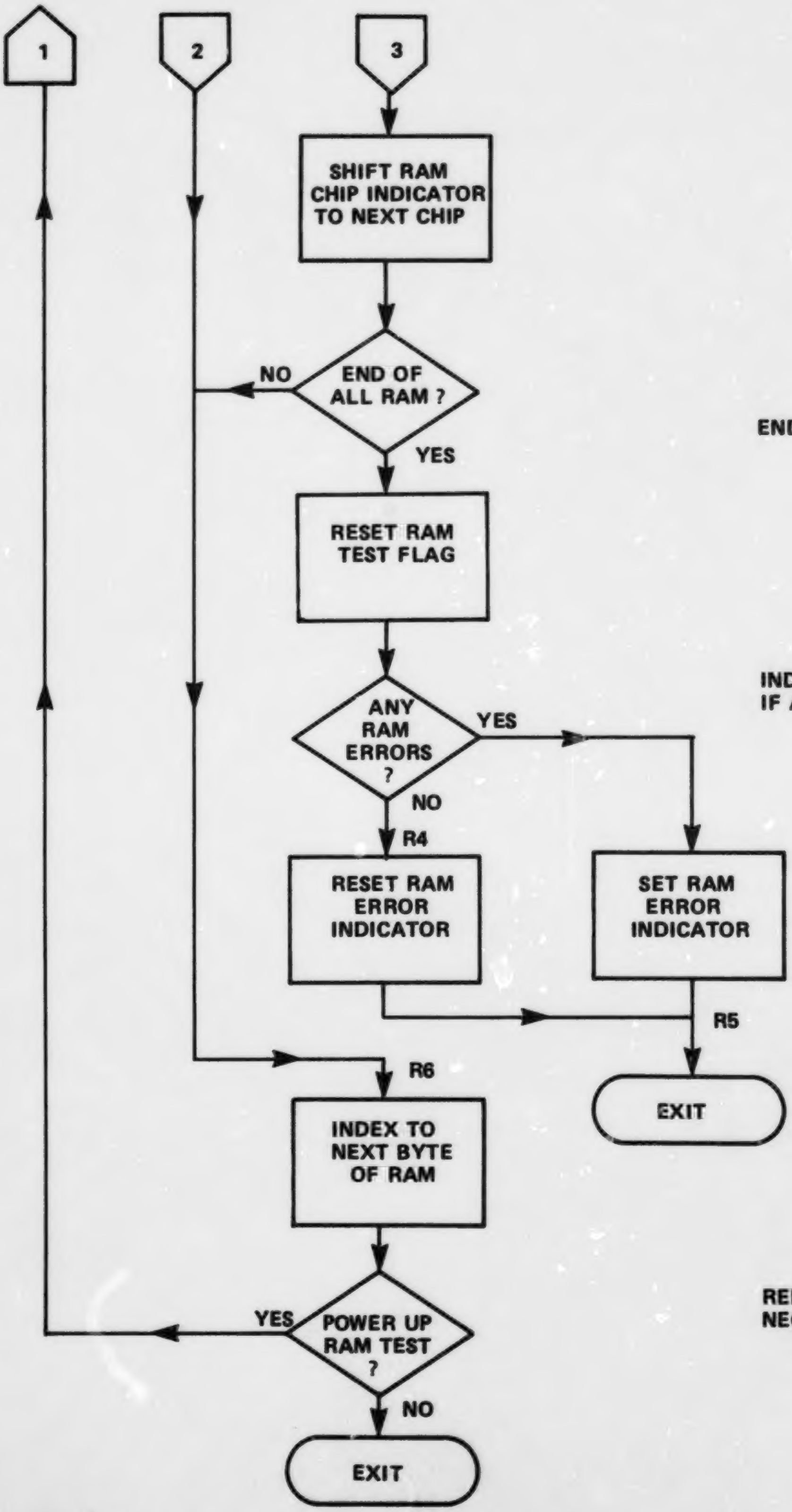
Page 1 of 2


ENTRY
RO
DISABLE INTERRUPTS
SAVE ORIGINAL DATA BYTE
TEST 1 BYTE OF RAM
RAM BYTE OK ?
NO
ERR
RESTORE ORIGINAL DATA
YES
RESTORE ORIGINAL DATA
ENABLE INTERRUPTS
ENABLE INTERRUPTS
SET RAM ERROR BIT FOR THIS CHIP
SKIP TO END OF THIS 2K CHIP
R2
NO
END OF 2K CHIP
YES
R3
1
2
3

TEST 1 BYTE OF RAM

SET ERROR BIT IF TEST FAILS

PERFORM RAM TEST ROUTINE (RAMCHK)


1
2
3
SHIFT RAM CHIP INDICATOR TO NEXT CHIP
END OF ALL RAM ?
NO
YES
RESET RAM TEST FLAG
ANY RAM ERRORS ?
YES
NO
R4
RESET RAM ERROR INDICATOR
SET RAM ERROR INDICATOR
R5
R6
EXIT
INDEX TO NEXT BYTE OF RAM
POWER UP RAM TEST ?
YES
NO
EXIT

END OF RAM

INDICATE ERROR IF ANY

REPEAT IF NECESSARY

7.10/ Power-Down Routine (IGNORE)

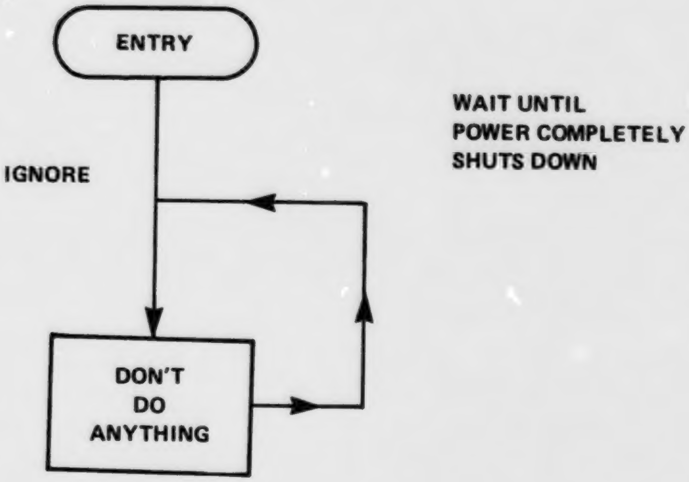
Purpose

This routine provides an orderly means of turning off the controller. It is executed whenever there is a power failure to the controller or when the controller is switched off.

Description

This routine merely branches to itself. After a power failure, the processor is capable of executing a few instructions, and this loop makes sure nothing destructive happens to RAM variables.

POWER DOWN ROUTINE (IGNORE) **Page 1 of 1**


ENTRY
IGNORE
DON'T DO ANYTHING
WAIT UNTIL POWER COMPLETELY SHUTS DOWN

Appendix A/ Communication Messages

APPENDIX A/ Communication Messages

The protocol established to effect communications (between the central computer and the field station controller) requires that messages adhere to certain formats. These formats are described in detail in this Appendix.

The central station always initiates a communication dialogue and the field station replies with a single response message.

There are 8 possible message codes (0-7), though only 6 are to be sent by the central station while all 8 can be sent by the field station. When the central station sends a message, the field station replies with the same message code, unless there is an error, in which case one of messages 0 or 2 is returned. The description of each message follows.

Message 0 -- Logical Error (Table 1)

The field station will send a logical error message in response to any message from the central station that is logically incorrect, such as a message with an invalid length or date.

This message should never be sent by the central station. In the unlikely event that it is sent, the field station will return a logical error message.

Message 1 -- Send Field Station Memory (Table 2)

This message instructs the field station to transmit the contents of a requested range of memory to the central station. If the message is correct, message 1 is returned to the central station with the requested data imbedded in the message. A logical error message (0) will be returned to the central station for the following reasons:

- message format not correct,
- requested memory partially outside of RAM,

- requested memory partially outside of PROM,
- requested memory completely outside both RAM and PROM.

Message 2 -- Negative Acknowledgement (Table 3)

This message will be returned by the field station in response to any message in which certain errors occur (hardware errors, incorrect station address, byte count not between 4 and 22, inclusive, or incorrect checksum).

This message should never be sent by the central station, otherwise a logical error message (0) is returned to the central station.

Message 3 -- Send Hourly Volume Data (Table 4)

This message instructs the field station to send the volume counts collected for a specific date. If the message is correct and the data are available, the requested information is sent to the central station in a message (3). Otherwise, the following conditions will cause a logical error message (0) to be returned to the central station:

- message format is not correct,
- data are not available for the given day.

Message 4 -- send Length Count and Speed Data (Table 5)

This message instructs the field station to send length count and vehicle speed data collected for a given date. If the message format is correct and both speed and length data have been collected for the requested day, a reply message (4) is returned to the central station with the requested data. Otherwise, a logical error message (0) is returned.

Message 5 -- Start Speed and Length Data Accumulation (Table 6)

This message instructs the field station to start the collection of vehicle speed and length data at time 00:00 on a particular date. If the message is correct, a return message (5) is sent to the central station. Otherwise, the following conditions result in a logical error message (0) being set to the central station:

- message format is not correct,
- date requested has already passed,
- date requested is not valid (Feb. 29, but not a leap year).

This message is not currently implemented by the application software. It was intended for a controller with very little memory. Currently, speed and length data are collected continuously.

Message 6 -- End Speed and Length Data Accumulations (Table 7)

This message instructs the field station to terminate the collection of vehicle speed and length data at 24:00 on the requested date. If the message is correct and the processing is successful, a return message (6) is sent to the central station. Otherwise, one of the following errors has occurred and a logical error message (0) is sent to the central station.

- Message format is not correct.

- Speed/length collection in progress but requested end date already passed.

- Speed/length collection not in progress and no request to start has been made.

- Speed/length collection not in progress and end date does not follow start date.

- Requested end date is invalid (Feb. 29, but not leap year).

As for message 5, this message is not currently implemented by the application software.

Message 7 -- Change Field Station Memory (Table 8)

This message instructs the field station to make changes to its RAM memory and provides the new memory values. If the message is correct and processing is successful, then a return message (7) containing the contents of the requested memory after the change is returned to the central station. Otherwise, a logical error message (0) is returned to the central station because one of the following errors has occurred:

- format of message is not correct,
- memory to be changed lies partially or completely outside of RAM,
- security code is not correct and is not provided (correctly) in message.

Note that the memory contents returned by the field station may not be the same as the requested changes since some locations are protected from changes.

Generalized Message Format

Certain parts of the communication messages are common to several or all types of messages. The meaning and format of these message components are described below. The generalized message format is shown in Table 9.

Transmission Code/Station Address

This is the first byte of every communication message, to and from the central station. It is an 8-bit quantity CCCAAAAA where CCC is the 3-bit message code and AAAAA is the 5-bit station address.

Byte Count to Checksum

This is the number of bytes in the message, from the first byte (transmission code/station address) to the checksum byte, inclusive. It is the second byte of every message sent by the central station. It is the second and third bytes (i.e., a 2-byte quantity) of every message sent from the field station.

Checksum

Every message sent to or from the central station has a checksum byte preceding the end characters. This byte is the "exclusive or" of the bytes of the message starting at the first byte and ending at the byte that precedes the checksum.

Error Code

This byte is used only in messages sent by the field station, and it immediately precedes the checksum byte. It reports certain errors to the central station, which are then reported to the PCS operator for appropriate action. The bit assignment is shown below (rightmost bit = least significant bit = bit 0). If the bit is set, the indicated error has been encountered.

Bit Number	Error
0	detector malfunction
1	PROM error
2	long power loss (greater than 255 minutes)
3	communication error
4	default loop spacing or dimensions supplied during initialization (not implemented as yet)
5	geometry or volume definition incorrect -- over-ridden during initialization (not implemented as yet)
6	RAM error
7	reserved for future use

Reset Control Byte

This byte is used only by messages sent by the central station, and it immediately precedes the checksum byte. It is used to reset certain errors, as outlined below.

Bit Number	Meaning
detector location bits	
0	0 for Direction 1, 1 for Direction 2
1	0 for upstream loop, 1 for downstream loop
2	lane number, 1st bit
3	lane number, 2nd bit
4	lane number, 3rd bit
action bits	
5	1 to clear detector described by bits 0-4, 0 otherwise
6	1 to clear power loss counter and alarms, 0 otherwise
7	reserved for future use

End Characters

Every message sent from the central station terminates with three end characters -- $AA, $55 and $55. Every message sent from the field station terminates with four end characters -- $AA, $55, $55 and $55.

Note that if a field station receives a message complete except for its end characters, the message will still be processed correctly.

Message Formats

The following tables describe the format of messages to and from the central station.

Table A1/ Logical Error Message

INQUIRY MESSAGE
CENTRAL TO FIELD

ANY LOGICALLY INCORRECT MESSAGE

- INCORRECT MESSAGE LENGTH
- INVALID DATE
- REQUESTED DATA UNAVAILABLE
- INVALID REQUEST
  - e.g. – A REQUEST TO STOP DATA COLLECTION AT A DATE THAT HAS ALREADY PASSED

RESPONSE MESSAGE
FIELD TO CENTRAL

1	000 - - - - -	MESSAGE CODE/ STATION ADDRESS
2	00000000	BYTE COUNT (MSB)
3	00000101	BYTE COUNT (LSB)
4		ERROR CODE
5		CHECKSUM
6	10101010	1ST END CHARACTER
7	01010101	2ND END CHARACTER
8	01010101	3RD END CHARACTER
9	01010101	4TH END CHARACTER

Table A2/ Send Field Station Memory Message

INQUIRY MESSAGE
CENTRAL TO FIELD

1	001-----	MESSAGE CODE/ STATION ADDRESS
2	00000110	BYTE COUNT
3		STARTING ADDRESS (MSB)
4		STARTING ADDRESS (LSB)
5		No. BYTES TO SEND*
6		CHECKSUM
7	10101010	1ST END CHARACTER
8	01010101	2ND END CHARACTER
9	01010101	3RD END CHARACTER

NORMAL RESPONSE MESSAGE
FIELD TO CENTRAL

1	001 - - - -	MESSAGE CODE/ STATION ADDRESS	
2		BYTE COUNT (MSB)	
3		BYTE COUNT (LSB)	
4		ADDRESS (MSB)	SAME AS INQUIRY MESSAGE
5		ADDRESS (LSB)	
6		NO. BYTES	
7		BYTE FROM MEMORY*	
8		BYTE FROM MEMORY	
	• • •	• • •	
		BYTE FROM MEMORY	
Y-1		BYTE FROM MEMORY	
Y		ERROR CODE	
Y+1		CHECKSUM	
Y+2	10101010	1ST END CHARACTER	
Y+3	01010101	2ND END CHARACTER	
Y+4	01010101	3RD END CHARACTER	
Y+5	01010101	4TH END CHARACTER	

* AT MOST 255 BYTES CAN BE SENT PER MESSAGE

Table A3/ Negative Acknowledgement Message

INQUIRY MESSAGE
CENTRAL TO FIELD

ANY MESSAGE WITH

- IMPROPER STATION ADDRESS
- CHARACTER TRANSMISSION ERROR

- INVALID MESSAGE LENGTH
- INCORRECT CHECKSUM

RESPONSE MESSAGE
FIELD TO CENTRAL

1	010-----	MESSAGE CODE/ STATION ADDRESS
2	00000000	BYTE COUNT (MSB)
3	00000101	BYTE COUNT (LSB)
4		ERROR CODE
5		CHECKSUM
6	10101010	1ST END CHARACTER
7	01010101	3RD END CHARACTER
8	01010101	3RD END CHARACTER
9	01010101	4th END CHARACTER

Table A4/ Send Hourly Volume Count Message

INQUIRY MESSAGE
CENTRAL TO FIELD

1	011 - - - - -	MESSAGE CODE/ STATION TO FIELD
2	00000111	BYTE COUNT
3		YEAR OF CENTURY (BCD)
4		MONTH OF YEAR (BCD)
5		DAY OF MONTH (BCD)
6		RESET CONTROL BYTE
7		CHECKSUM
8	10101010	1ST END CHARACTER
9	01010101	2ND END CHARACTER
10	01010101	3RD END CHARACTER

NORMAL RESPONSE MESSAGE
FIELD TO CENTRAL

1	011- - - - -	MESSAGE CODE/ STATION ADDRESS	
2		BYTE COUNT (MSB)	
3		BYTE COUNT (LSB)	
4		YEAR	SAME AS INQUIRY MESSAGE
5		MONTH	
6		DAY	
7		DAY OF WEEK	
8		VOLUME DATA	
	⋮	⋮	
		VOLUME DATA	
Y−1		VOLUME DATA	
Y		ERROR CODE	
Y+1		CHECKSUM	
Y+2	10101010	1ST END CHARACTER	
Y+3	01010101	2ND END CHARACTER	
Y+4	01010101	3RD END CHARACTER	
Y+5	01010101	4th END CHARACTER	

Table A5/ Send Length Count and Speed Data Message

INQUIRY MESSAGE
CENTRAL TO FIELD

1	100-----	MESSAGE CODE/ STATION ADDRESS
2	00000111	BYTE COUNT
3		YEAR OF CENTURY (BCD)
4		MONTH OF YEAR (BCD)
5		DAY OF MONTH (BCD)
6		RESET CONTROL BYTE
7		CHECKSUM
8	10101010	1ST END CHARACTER
9	01010101	2ND END CHARACTER
10	01010101	3RD END CHARACTER

NORMAL RESPONSE MESSAGE
FIELD TO CENTRAL

1	100-----	MESSAGE CODE/ STATION ADDRESS	
2		BYTE COUNT (MSB)	
3		BYTE COUNT (LSB)	
4		YEAR	SAME AS INQUIRY MESSAGE
5		MONTH	
6		DAY	
7		DAY OF WEEK	
8		LENGTH DATA	
	⋮	⋮	
		LENGTH DATA	
		YEAR	SAME AS INQUIRY MESSAGE
		MONTH	
		DAY	
		DAY OF WEEK	
		SPEED DATA	
	⋮	⋮	
		SPEED DATA	
Y-1		SPEED DATA	
Y		ERROR CODE	
Y+1		CHECKSUM	
Y+2	10101010	1ST END CHARACTER	
Y+3	01010101	2ND END CHARACTER	
Y+4	01010101	3RD END CHARACTER	
Y+5	01010101	4TH END CHARACTER	

Table A6/ Start Speed and Length Data Accumulation Message

INQUIRY MESSAGE
CENTRAL TO FIELD

1	101- - - - -	MESSAGE CODE/ STATION ADDRESS
2	00000111	BYTE COUNT
3		YEAR OF CENTURY (BCD)
4		MONTH OF YEAR (BCD)
5		DAY OF MONTH (BCD)
6		RESET CONTROL BYTE
7		CHECKSUM
8	10101010	1ST END CHARACTER
9	01010101	2ND END CHARACTER
10	01010101	3RD END CHARACTER

NORMAL RESPONSE MESSAGE
FIELD TO CENTRAL

1	101- - - - -	MESSAGE CODE/ STATION ADDRESS	
2	00000000	BYTE COUNT (MSB)	
3	00001000	BYTE COUNT (LSB)	
4		YEAR	SAME AS INQUIRY MESSAGE
5		MONTH	
6		DAY	
7		ERROR CODE	
8		CHECKSUM	
9	10101010	1ST END CHARACTER	
10	01010101	2ND END CHARACTER	
11	01010101	3RD END CHARACTER	
12	01010101	4TH END CHARACTER	

Table A7/End Speed and Length Data Accumulation Message

INQUIRY MESSAGE
CENTRAL TO FIELD

1	110- - - - -	MESSAGE CODE/ STATION ADDRESS
2	00000111	BYTE COUNT
3		YEAR OF CENTURY (BCD)
4		MONTH OF YEAR (BCD)
5		DAY OF MONTH (BCD)
6		RESET CONTROL BYTE
7		CHECKSUM
8	10101010	1ST END CHARACTER
9	01010101	2ND END CHARACTER
10	01010101	3RD END CHARACTER

NORMAL RESPONSE MESSAGE
FIELD TO CENTRAL

1	110- - - - -	MESSAGE CODE/ STATION ADDRESS	
2	00000000	BYTE COUNT (MSB)	
3	00001000	BYTE COUNT (LSB)	
4		YEAR	SAME AS INQUIRY MESSAGE
5		MONTH	
6		DAY	
7		ERROR CODE	
8		CHECKSUM	
9	10101010	1ST END CHARACTER	
10	01010101	2ND END CHARACTER	
11	01010101	3RD END CHARACTER	
12	01010101	4TH END CHARACTER	

Table A8/Change Field Station Memory Message

INQUIRY MESSAGE
CENTRAL TO FIELD

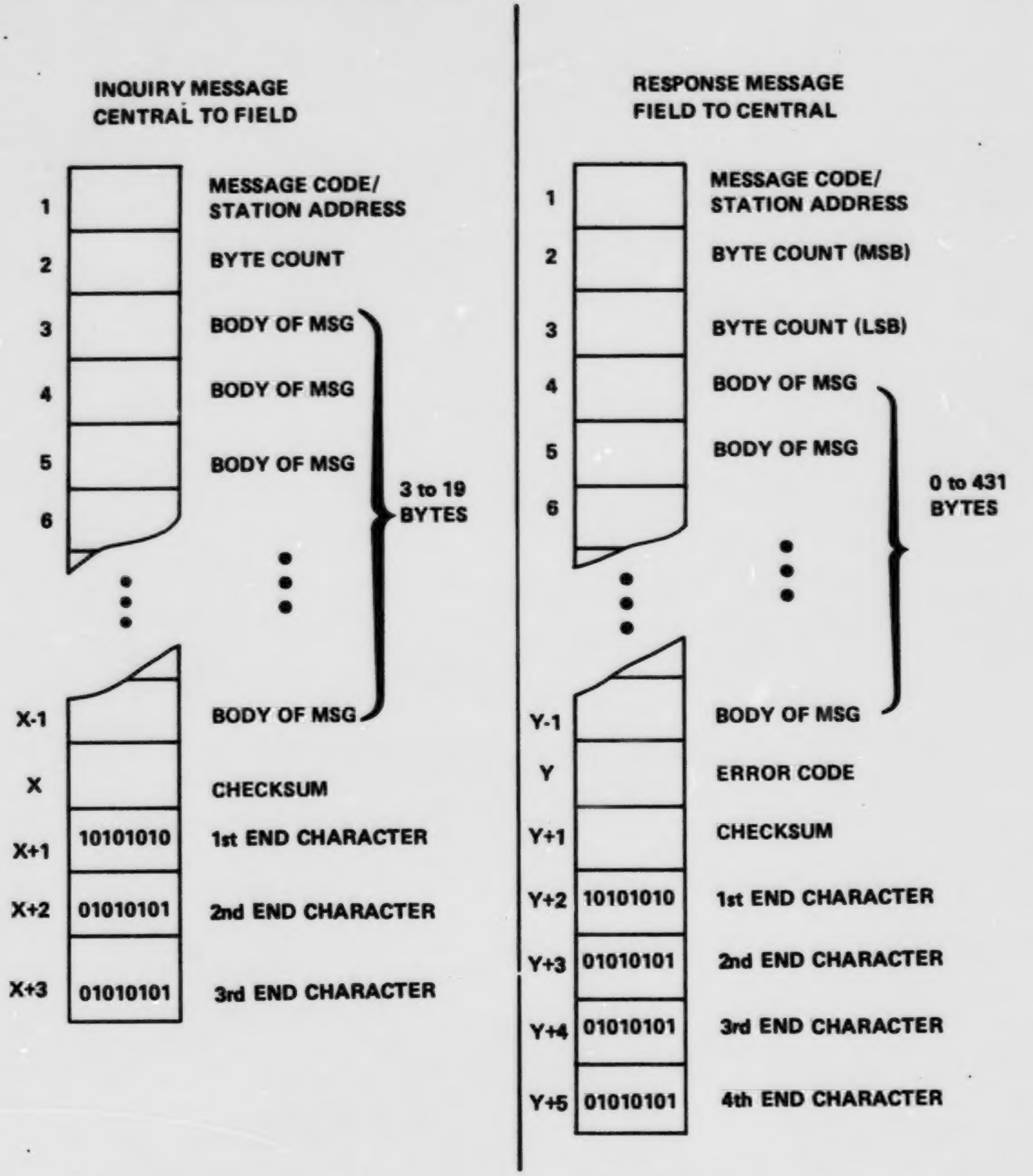
1	111-----	MESSAGE CODE/ STATION ADDRESS
2		BYTE COUNT
3		STARTING ADDRESS OF CHANGE (MSB)
4		STARTING ADDRESS OF CHANGE (LSB)
5		No. BYTES TO CHANGE*
6		NEW MEMORY DATA
7		NEW MEMORY DATA
	⋮	⋮
		NEW MEMORY DATA
X-1		NEW MEMORY DATA
X		CHECKSUM
X+1	10101010	1ST END CHARACTER
X+2	01010101	2ND END CHARACTER
X+3	01010101	3RD END CHARACTER

NORMAL RESPONSE MESSAGE
FIELD TO CENTRAL

1	111-----	MESSAGE CODE/ STATION ADDRESS	
2		BYTE COUNT (MSB)	
3		BYTE COUNT (LSB)	
4		ADDRESS (MSB)	SAME AS INQUIRY MESSAGE
5		ADDRESS (LSB)	
6		No. BYTES	
7		MEMORY DATA AFTER CHANGE	
8		MEMORY DATA "	
	⋮	⋮	
		MEMORY DATA "	
Y-1		MEMORY DATA "	
Y		ERROR CODE	
Y+1		CHECKSUM	
Y+2	10101010	1ST END CHARACTER	
Y+3	01010101	2ND END CHARACTER	
Y+4	01010101	3RD END CHARACTER	
Y+5	01010101	4TH END CHARACTER	

* AT MOST 16 BYTES CAN BE CHANGED WITH 1 MESSAGE

Table A9/ Generalized Message Format


INQUIRY MESSAGE
CENTRAL TO FIELD
1
MESSAGE CODE/ STATION ADDRESS
2
BYTE COUNT
3
BODY OF MSG
4
BODY OF MSG
5
BODY OF MSG
6
3 to 19 BYTES
X-1
BODY OF MSG
X
CHECKSUM
X+1
10101010
1st END CHARACTER
X+2
01010101
2nd END CHARACTER
X+3
01010101
3rd END CHARACTER
RESPONSE MESSAGE
FIELD TO CENTRAL
1
MESSAGE CODE/ STATION ADDRESS
2
BYTE COUNT (MSB)
3
BYTE COUNT (LSB)
4
BODY OF MSG
5
BODY OF MSG
6
0 to 431 BYTES
Y-1
BODY OF MSG
Y
ERROR CODE
Y+1
CHECKSUM
Y+2
10101010
1st END CHARACTER
Y+3
01010101
2nd END CHARACTER
Y+4
01010101
3rd END CHARACTER
Y+5
01010101
4th END CHARACTER

APPENDIX B/ Setting Up of the PROM Checksum Tables

APPENDIX B/ Setting Up of the PROM Checksum Tables

The PROM checksum (defined as 16 least significant bits of the sum of all data bytes) is calculated for every 2K EPROM chip in turn and compared with the value stored in the table.

The checksum table is located just past the subroutine CALPRO at the hex location $FFD6. It contains nine 2-byte long entries. Eight of those are used for storing the checksums for the eight possible chips and the last one contains a compensating value required to "fix" the final checksum.

The routine calculates the checksums for all eight possible chips and, consequently, it is necessary to supply values for empty chip locations ($F800; as Topping hardware reads all empty locations as ones). The checksum values for the other chips are simply obtained from the DATA I/O programmer, except for chip 8 ($F800) which contains the checksum table. In this case, the final checksum value and the compensating value have to be calculated so the checksum calculated will be equal to the checksum stored in the table.

In order to calculate the correct values it is first necessary to calculate the checksum for this chip with the checksums for the first seven chips in the table but with the final checksum-plus-compensating-value equal to zero (see Figure B-1).

If the value read from the PROM programmer is n (2 bytes), the actual checksum for this chip can be represented as:

$$\begin{array}{lll} k * 65536 + n & \text{where} & A = B = C = D = 0 \\ & & k = \text{integer} \\ & & o < n < 65535 \end{array}$$

If the values of A, B, C and D are non-zero, the checksum can be represented as:

$$k * 65536 + n + A + B + C + D$$

or

$$x * 65536 + 256\ A + B$$

where k and x are integers and the value $x * 65536$ is the most significant part.

$x > k$ but max difference can be only 1 as $4 \times 255 < 65536$

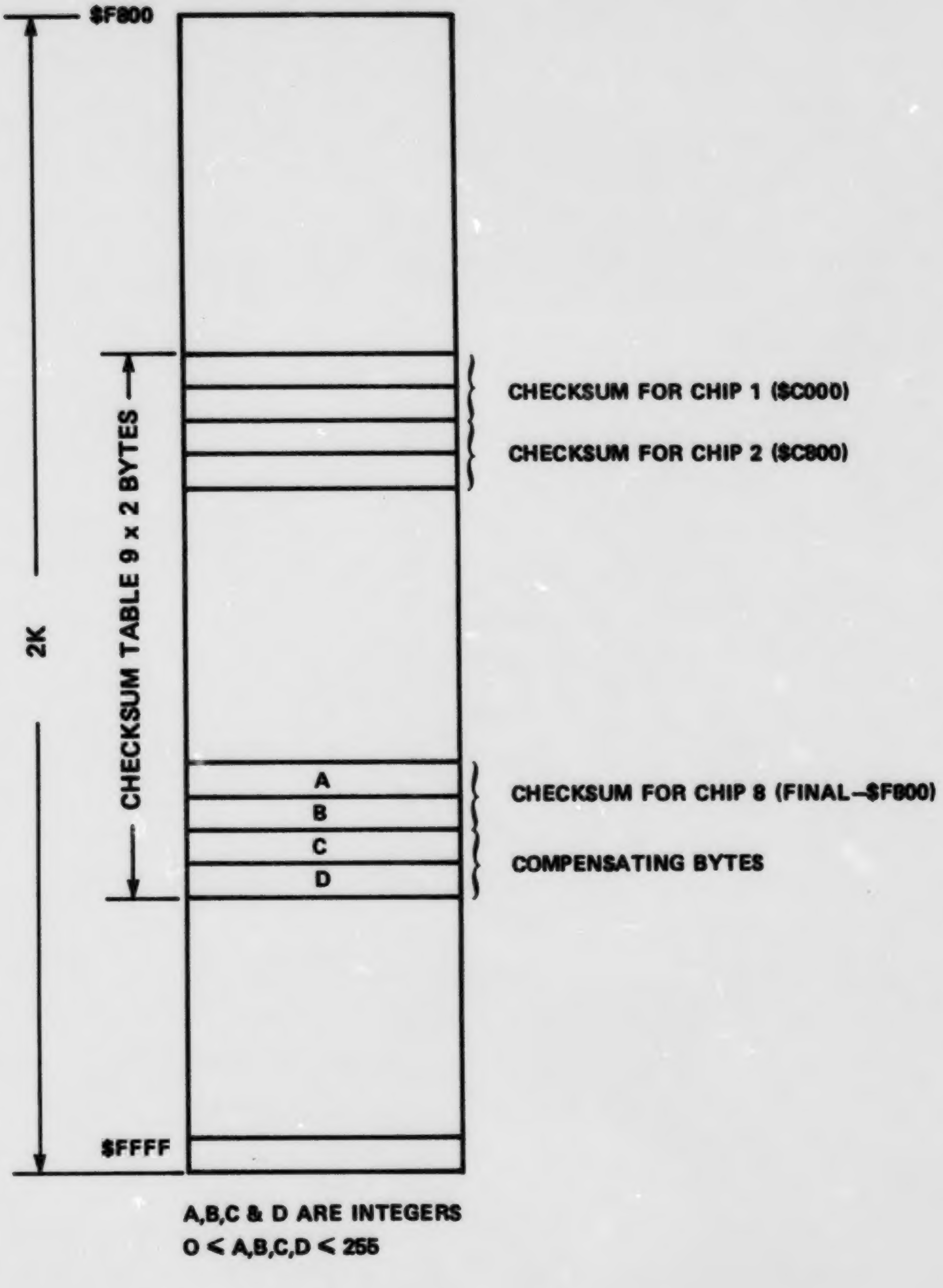

$F800
2K
CHECKSUM TABLE 9 x 2 BYTES
CHECKSUM FOR CHIP 1 ($C000)
CHECKSUM FOR CHIP 2 ($C800)
A
B
C
D
CHECKSUM FOR CHIP 8 (FINAL–$F800)
COMPENSATING BYTES
$FFFF
A,B,C & D ARE INTEGERS
0 < A,B,C,D < 255

Figure B1/ Checksum Calculations – Chip Containing Checksum Table

Now $k * 65536 + n + A + B + C + D = x * 65536 + 256\ A + B$

or $65536\ (k - x) + n + (C + D) = 255A$

Hence, $A = \dfrac{65536\ (k - x) + n + C + D}{255}$

Case 1

$$x - k = 0$$

$$A = \frac{n + C + D}{255} \qquad 0 < A < 255$$

for $A < 255 \rightarrow n + C + D < 65025$

or $n < 65025$ for $C = D = 0$

Case 2

$$x - k = -1$$

$$A = \frac{-\ 65536 + n + C + D}{255}$$

If $A = 0 \rightarrow n + C + D = 65536$

$A = 1 \rightarrow n + C + D = 65791$

$A = 2 \rightarrow n + C + D = 66556 = 66046 + 510$ (impossible)

Hence, the following formulae apply:

1/ $n < 65025$ (\$FE01)

$A = \dfrac{n + C + D}{255}$ choose C & D to make A integer
B arbitrary

2/ $n > 65025$ (\$FE01)

$A = 0$ choose $C + D = 65536 - n$
B arbitrary

Now $k * 65536 + n + A + B + C + D = x * 65536 + 256\,A + B$

or $65536\,(k - x) + n + (C + D) = 255A$

Hence, $A = \dfrac{65536\,(k - x) + n + C + D}{255}$

Case 1

$$x - k = 0$$

$$A = \frac{n + C + D}{255} \qquad 0 < A < 255$$

for $A < 255 \rightarrow n + C + D < 65025$

or $n < 65025$ for $C = D = 0$

Case 2

$$x - k = -1$$

$$A = \frac{-\,65536 + n + C + D}{255}$$

If $A = 0 \rightarrow n + C + D = 65536$
$A = 1 \rightarrow n + C + D = 65791$
$A = 2 \rightarrow n + C + D = 66556 = 66046 + 510$ (impossible)

Hence, the following formulae apply:

1/ $n < 65025$ (\$FE01)

$A = \dfrac{n + C + D}{255}$ choose C & D to make A integer
B arbitrary

2/ $n > 65025$ (\$FE01)

$A = 0$ choose $C + D = 65536 - n$
B arbitrary

APPENDIX C/ Program Listings

APPENDIX C/ PROGRAM LISTINGS

Page

1/ Equate File (EQUATE) 227

2/ Communications Routines. 242
  2.1/ Main Routine (COMM) 242
  2.2/ Receiving Routine (RECV) 245
  2.3/ Transmit First Character Routine (TFIRST) 250
  2.4/ Transmit Routine (XMIT) 251
  2.5/ Prepare Answer Routine (PREANS) 255
  2.6/ Communications Timer Routine (TIMER1) 278
  2.7/ Security Code Timer Routine (TIMER2). 281

3/ Master Initialization Routine (MASINT) 282

4/ RAM Test Routines (RAM). 284

5/ Keyboard Routine (KEYBD) 288

6/ Clock Routine (CLOCK). 304

7/ Power-Up Routine (POWRON). 314

8/ Executive Routine (EXEC) 324

9/ Compute PROM Checksum Routine (CALPRO) 328

10/ Power Down Routine (IGNORE) 334

```
PAGE 001  EQUATE  .SA:1

                      OPT    LLE=120
                  *
                  *     EQUATE FILE USED FOR PERMANENT COUNTING STATION PROGRAMS
                  *
                  *
                  *       ******************************************************
                  *       *                                                    *
                  *       *  PROGRAMMER:   W.J. WIERCIENSKI                    *
                  *       *                                                    *
                  *       *  MINISTRY OF TRANSPORTATION AND COMMUNICATIONS     *
                  *       *  TRANSPORTATION TECHNOLOGY AND ENERGY              *
                  *       *  1201 WILSON AVENUE                                *
                  *       *  DOWNSVIEW, ONTARIO  M3M 1J8                       *
                  *       *                                                    *
                  *       *  TELEPHONE (416) 248-7141                          *
                  *       *                                                    *
                  *       *  NOVEMBER  1982 AD                                 *
                  *       *                                                    *
                  *       ******************************************************
                  *
                  *
```

```
                 *
                 *
                 *        PAGE 0 COLUMN 0    MAIN VARIABLES
                 *
                 *
0000  A PCSCAD EQU    $00      PCS CONTROLLER ADDRESS VALUE (0 TO 31) - 5 BITS , RIGHT JUSTIFIED
0001  A SECCOD EQU    $01      SECURITY CODE  (= 0705 - 2 BYTES) PROTECTS RAM FROM UNAUTHORIZED CHANGES
0003  A DATE   EQU    $03      CURRENT DATE, 3 BYTES - (YEAR, MONTH, DAY)
0003  A YEAR   EQU    $03      CURRENT CALENDAR YEAR OF CENTURY  (BCD)
0004  A MONTH  EQU    $04      CURRENT CALENDAR MONTH  (BCD)
0005  A DAOFMO EQU    $05      CURRENT DAY OF MONTH  (BCD)
0006  A DOYM   EQU    $06      CURRENT DAY OF YEAR - MSB  (BCD)
0007  A DOYL   EQU    $07      CURRENT DAY OF YEAR - LSB  (BCD)
                 *
0008  A DAOFWK EQU    $08      DAY OF WEEK (1= MONDAY)
0009  A HR     EQU    $09      HOUR OF DAY - 24 HR CLOCK  (BCD: 0-23)
000A  A MIN    EQU    $0A      CURRENT MINUTE  (BCD: 0-59)
000B  A SEC    EQU    $0B      CURRENT SECOND  (BCD: 0-59)
000C  A TSEC   EQU    $0C      1/10 SECOND COUNTER (0-9)
                 *
                 *        THE FOLLOWING VARIABLES ARE UPDATED PERIODICALLY -
                 *        NO MANUAL INITIALIZATION IS NECESSARY
                 *
000D  A DIM    EQU    $0D      NUMBER OF DAYS IN CURRENT MONTH (BCD)
000E  A DAYPTR EQU    $0E      BIT POINTER FOR DAY-OF-WEEK  (LSB=MON.)
000F  A TIMEFX EQU    $0F      TIME CORRECTION FLAG-TO PERFORM DAYLIGHT
                 *             SAVING CHANGE (LAST SUNDAY APRIL/OCTOBER 2AM)
                 *             =0  CHANGE ALREADY DONE
                 *             >0  CHANGE TO BE DONE
                 *
```

```
PAGE  003  EQUATE  .SA:1

                 *
                 *
                 *
                 *          PAGE 0 COLUMN 1    STATION GEOMETRY & VOLUME DESCRIPTION VARIABLES
                 *
                 *
       0010  A NLANE1 EQU    $10      NUMBER OF LANES EQUIPPED WITH DETECTOR(S) IN DIR. #1(N/B OR W/B) - MAX 8
       0011  A NLANE2 EQU    $11      NUMBER OF LANES EQUIPPED WITH DETECTOR(S) IN DIR. #2(S/B OR E/B) - MAX 8
       0012  A LDESW1 EQU    $12      UPSTREAM LANE DESCRIPTION BYTE IN DIRECTION #1.
                 *                    BIT 0 = 1 FOR LANE 1 DETECTOR ETC.IF 0 NO DETECTOR OR LANE EXISTS.
       0013  A LDESW2 EQU    $13      SAME AS LDESW1,BUT FOR DIRECTION #2.
       0014  A LDNW1  EQU    $14      DOWNSTREAM LANE DESCRIPTOR BYTE IN DIRECTION #1
       0015  A LDNW2  EQU    $15      SAME AS LDNW1,BUT FOR DIRECTION #2
       0016  A NVOL1  EQU    $16      NUMBER OF VOLUME COUNTS TO BE CALCULATED IN DIRECTION #1 - MAX 8
       0017  A NVOL2  EQU    $17      NUMBER OF VOLUME COUNTS TO BE CALCULATED IN DIRECTION #2 - MAX 8
                 *                    MAX TOTAL OF VOLUME COUNTS FOR BOTH DIRECTIONS EQUALS 8
       0018  A VOLDW1 EQU    $18      VOLUME COUNT DESCRIPTOR BYTE FOR VOLUME #1.
                 *                    LANE BIT=1 MEANS THAT THIS LANE CONTRIBUTES TO THIS VOLUME.
                 *                    LANES NUMBERED RIGHT TO LEFT IE BIT 0 = LANE 1(CENTRE).
                 *                    VOLUME DESCRIPTION STARTS WITH DIRECTION #1
                 *                    IMMEDIATELY FOLLOWED BY DIRECTION #2.
                 *                    THE UNUSED VOLUME DESCRIPTION WORDS WILL BE FILLED WITH BINARY ZEROES.
       0019  A VOLDW2 EQU    $19      VOLUME COUNT DESCRIPTOR BYTE FOR VOLUME #2
       001A  A VOLDW3 EQU    $1A      AS ABOVE FOR VOLUME #3
                 *
       001B  A VOLDW4 EQU    $1B      AS ABOVE FOR VOLUME #4
       001C  A VOLDW5 EQU    $1C      AS ABOVE FOR VOLUME #5
       001D  A VOLDW6 EQU    $1D      AS ABOVE FOR VOLUME #6
       001E  A VOLDW7 EQU    $1E      AS ABOVE FOR VOLUME #7
       001F  A VOLDW8 EQU    $1F      AS ABOVE FOR VOLUME #8
```

```
PAGE  004  EQUATE  .SA:1

                     *
                     *
                     *           PAGE 0 COLUMN 2    VARIABLES REQUIRED BY FRONT PANEL SERVICE ROUTINE
                     *
                     *
          0020   A KYSTAT EQU    $20      CURRENT STATE OF KEYBOARD (SEE KEYBD PROGRAM)
                     *
                     *           DISPLAY BUFFER
                     *
          0021   A DISP1  EQU    $21      DISPLAY BUFFER - MINUTES POSITION (MSB)
          0022   A DISP2  EQU    $22      DISPLAY BUFFER - MINUTES POSITION (LSB)
                     *
          0023   A DISP3  EQU    $23      DISPLAY BUFFER - SECONDS POSITION (MSB)
          0024   A DISP4  EQU    $24      DISPLAY BUFFER - SECONDS POSITION (LSB)
                     *
          0025   A DISP5  EQU    $25      DISPLAY BUFFER - HOURS POSITION (MSB)
          0026   A DISP6  EQU    $26      DISPLAY BUFFER - HOURS POSITION (LSB)
                     *
          0027   A LITES  EQU    $27      LIGHTS 0-7 (BIT0-LIGHT0, ..., BIT7-LIGHT7)
          0028   A LITES2 EQU    $28      LIGHTS 8-9 (BIT0-LIGHT8, BIT1=LIGHT9)
                     *
                     *
                     *
          0029   A LSTKEY EQU    $29      LAST KEY PRESS INDICATOR
                     *                     =0 - NO KEY WAS BEING PUSHED LAST TIME KEYBD WAS CALLED
                     *                     >0 - OTHERWISE
          002A   A ADDRSS EQU    $2A      ADDRESS ENTERED VIA KEYBOARD (2 BYTES)
          002C   A DISTMR EQU    $2C      DISPLAY TIMER
                     *                     UPDATED BY CLOCK S/R ON 30 SEC MARK TO RESET KEYBOARD
                     *                     255 MIN AFTER LAST KEY WAS PRESSED
                     *                     - ALSO USED DURING POWER UP TO BRIEFLY DISPLAY RESULTS
                     *                     OF PROM CHECKSUM CALCULATIONS
          002D   A DSPMSK EQU    $2D      DISPLAY MASK - BLANKS OUT UPPER AND/OR LOWER DISPLAY
                     *                     =$20  - BLANKS OUT UPPER DISPLAY
                     *                     =$60  - BLANKS OUT BOTH DISPLAYS
          002E   A RAMERR EQU    $2E      RAM ERROR BIT MAP (2 BYTES)
                     *                    BIT 0 OF RAMERR SET IF RAM CHIP #1 IS BAD ($0000-$07FF)
                     *                    ...
                     *                    BIT 7 OF RAMERR SET IF RAM CHIP #8 IS BAD ($3800-$3FFF)
                     *                    BIT 0 OF RAMERR+1 SET IF RAM CHIP #9 IS BAD ($4000-$47FF)
                     *                    BIT 1 OF RAMERR+1 SET IF RAM CHIP #10 IS BAD ($4800-$4FFF)
```

```
PAGE  005  EQUATE  .SA:1

                          *
                          *
                          *          PAGE 0 COLUMN 3    TEMPORARY COUNTS,SAVE AREAS,FLAGS ETC
                          *
                          *
              0030   A PUCR    EQU     $30       60 HZ PULSE COUNT (0-5)
              0031   A SSEC    EQU     $31       6 SEC COUNT (0-5)
              0032   A SMIN    EQU     $32       6 MIN COUNT (0-5)
              0033   A TMFLG1  EQU     $33       TIME FLAGS < 1 MIN
                          *                      $01  -  60 HZ FLAG
                          *                      $02  -  30 HZ
                          *                      $04  -  20 HZ
                          *                      $08  -  10 HZ
                          *                      $10  -  1/2 HZ
                          *                      $20  -  1 SEC
                          *                      $40  -  6 SEC
                          *                      $80  -  30 SEC
                          *
              0034   A TMFLG2  EQU     $34       TIME FLAGS => 1 MIN
                          *                      $01  -  1 MIN FLAG
                          *                      $02  -  6 MIN
                          *                      $04  -  15 MIN
                          *                      $08  -  20 MIN
                          *                      $10  -  30 MIN
                          *                      $20  -  1 HR
                          *                      $40  -  24 HRS
                          *                      $80  -  24 HRS
                          *
              0035   A EXFLG   EQU     $35       TIME FLAG SAVE AREA
              0036   A DWNCNT  EQU     $36       POWER FAILURE COUNTER
              0037   A ALARMS  EQU     $37       CONTROLLER ERROR ALARMS
                          *                      02 - POWERDOWN TIME > 255 MIN
                          *                      80 - POWERDOWN BIT
                          *
              0038   A EXINDX  EQU     $38       INDEX REG SAVE AREA (2 BYTES)
                          *
              003A   A SBRINX  EQU     $3A       PGM-TO-BE-EXECUTED ADDRESS SAVE AREA (2 BYTES)
              003C   A TEMP06  EQU     $3C       TEMP LOCATION USED BY KEYBD ROUTINE
              003D   A TEMP07  EQU     $3D       TEMP LOCATION USED BY KEYBD ROUTINE
              003E   A TEMP01  EQU     $3E       TEMP LOCATION USED BY COMM ROUTINE AND
                          *                       CLOCK ROUTINE (MUTUALLY EXCLUSIVE ROUTINES)
              003F   A TEMP02  EQU     $3F       TEMP LOCATION USED BY COMM ROUTINE
```

```
PAGE  006  EQUATE  .SA:1

                      *
                      *
                      *     PAGE 0 COLUMN 4    VARIABLES REQUIRED BY COMMUNICATIONS ROUTINE
                      *
                      *
         0040  A TCODE  EQU    $40      SAVE AREA FOR MSG CODE (LEFT JUSTIFIED) OR INITIAL MSG BYTE
                      *                 LEFT 3 BITS = TRANSMISSION CODE
                      *                 RIGHT 5 BITS = STATION ADDRESS
         0041  A MSGLEN EQU    $41      LENGTH OF RETURN MSG (2 BYTES) - INCLUDES MSG CODE BYTE AND CHECKSUM BYTE
         0043  A CHECKS EQU    $43      CHECKSUM (CALCULATED)
         0044  A ERCODE EQU    $44      ERROR CODE WORD - TO BE TRANSMITTED BACK TO NOVA
                      *                 BIT 0 = DETECTOR MALFUNCTION
                      *                 BIT 1 = PROM CHECKSUM ERROR
                      *                 BIT 2 = TIME-OF-DAY CLOCK ERROR (POWERDOWN > 255 MINUTES)
                      *                 BIT 3 = COMMUNICATIONS ERROR (PARITY OR CHECKSUM ERROR)
                      *                 BIT 4 = DEFAULT LOOP SPACING OR DIMENSION SUPPLIED BY
                      *                         MASTER INITIALIZATION (NOT IMPLEMENTED)
                      *                 BIT 5 = GEOMETRY OR VOLUME DEFINITION OVER-RIDDEN BY
                      *                         MASTER INITIALIZATION (NOT IMPLEMENTED)
                      *                 BIT 6 = RAM ERROR
                      *                 WHEN NO BITS ARE SET THERE ARE NO ERRORS
         0045  A RBITS  EQU    $45      RECEIVE INDICATOR
                      *                 = 0 NOT IN RECEIVING MODE (TRANSMITTING)
                      *                 = 1 BEGINNING OF MSG EXPECTED
                      *                 = 2 "# OF BYTES TO CHECKSUM" EXPECTED
                      *                 = 3 MSG TEXT BYTE EXPECTED
                      *                 = 4 CHECKSUM EXPECTED
                      *                 = 5 1-ST END CHARACTER EXPECTED
                      *                 = 6 2-ND END CHARACTER EXPECTED
                      *                 = 7 3-RD END CHARACTER EXPECTED
                      *                 > 7 SEARCHING FOR END OF MSG AFTER CATASTROPHIC FAILURE
                      *                 = 8 SEARCHING FOR FIRST END CHARACTER AFTER FAILURE
                      *                 = 9 SEARCHING FOR SECOND END CHARACTER AFTER FAILURE
                      *                 = 10 SEARCHING FOR THIRD END CHARACTER AFTER FAILURE
         0046  A XBITS  EQU    $46      TRANSMITTING INDICATOR
                      *                 = 0 IN RECEIVING MODE
                      *                 = 1 SEND LENGTH BYTE #1
                      *                 = 2 SEND LENGTH BYTE #2
                      *                 = 3 SEND DATA FROM BUFFER(S)
                      *                 = 4 SEND ERROR CODE BYTE
                      *                 = 5 SEND CHECKSUM BYTE
                      *                 = 6 SEND 1-ST END CHARACTER
                      *                 = 7 SEND 2-ND END CHARACTER
                      *                 = 8 SEND 3-RD END CHARACTER
                      *                 = 9 SEND 4-TH END CHARACTER
                      *                 = 10 LAST END CHARACTER SENT
         0047  A DPOINT EQU    $47      POINTER TO NEXT EMPTY SPACE IN INPUT BUFFER
                      *                 OR NEXT BYTE TO BE SENT FROM OUTPUT BUFFER AS APPLICABLE
         0049  A DLNGTH EQU    $49      NUMBER OF BYTES OF THE BODY OF THE MSG STILL TO BE
                      *                 RECEIVED (INPUT) OR TRANSMITTED (OUTPUT) AS APPLICABLE
         004B  A POINT2 EQU    $4B      POINTER TO DATA BUFFER #2 OR ZEROES IF NO DATA PRESENT (2 BYTES)
         004D  A LNGTH2 EQU    $4D      LENGTH OF DATA IN BUFFER #2 OR ZEROES IF NO DATA PRESENT (2 BYTES)
         004F  A CERROR EQU    $4F      NUMBER OF ERRORS (PARITY,OVERRUN,FRAMING ETC) SINCE
                      *                 LAST ERASING OR MASTER INITIALIZATION
```

```
PAGE  007  EQUATE  .SA:1

00226                      *
00227                      *
00228                      *          PAGE 0 COLUMN 5    INPUT/OUTPUT AREA USED BY COMMUNICATIONS ROUTINE
00229                      *
00230                      *
00231             0050   A INDATA EQU    $50      BEGINNING OF INPUT/OUTPUT BUFFER (19 BYTES)
00232                      *
00233                      *
```

```
PAGE  008  EQUATE  .SX:1

                         *
                         *
                         *         PAGE 0 COLUMN 6    MISCELANEOUS VARIABLES
                         *
                         *
          0063  A CTIME1 EQU    $63      COMMUNICATIONS ROUTINE TIMER 1 - TIMES LENGTH OF INPUT MSG
                         *               (1/10 SEC INTERVALS); USED TO BREAK DEADLOCK IF INPUT TAKES TOO LONG.
          0064  A CTIME2 EQU    $64      COMMUNICATIONS ROUTINE TIMER 2 - TIMES LENGTH OF GAP BETWEEN CHARACTERS
                         *               DURING TRANSMISSION; USED TO BREAK DEADLOCK IF DATA CARRIER LOST
                         *               ( INTERRUPT MASKED DURING TRANSMISSION )
          0065  A CTIME3 EQU    $65      TIMER USED BY COMMUNICATIONS ROUTINE TO WIPE OUT SECURITY CODE
                         *
                         *         DATE WHEN SPEED & LENGTH CALCULATIONS ARE TO BE INITIATED
                         *
          0066  A SDATE  EQU    $66      STARTING DATE (YEAR, MONTH, DAY)
          0066  A SYEAR  EQU    $66      YEAR
          0067  A SMONTH EQU    $67      MONTH
          0068  A SDAY   EQU    $68      DAY
                         *
                         *         DATE WHEN SPEED & LENGTH CALCULATIONS ARE TO BE TERMINATED
                         *
          0069  A EDATE  EQU    $69      ENDING DATE (YEAR, MONTH, DAY)
          0069  A EYEAR  EQU    $69      YEAR
          006A  A EMONTH EQU    $6A      MONTH
          006B  A EDAY   EQU    $6B      DAY
                         *
          006C  A MFLAG  EQU    $6C      FLAGS
                         *               BIT 0 - START SPEED CALCULATION AT "0000" HOUR ON DAY REQUESTED
                         *               BIT 1 - STOP SPEED CALCULATION AT "2400" HOUR ON DAY REQUESTED
                         *               BIT 2 TO BIT 3 - RESERVED FOR FUTURE USE
                         *               BIT 4 - SECURITY CODE TIMER
                         *               BIT 5 - PERFORM PROM CHECKSUM CALCULATIONS
                         *               BIT 6 - PERFORM A RAM TEST
                         *               BIT 7 - RESERVED FOR FUTURE USE
          006D  A MBYTE  EQU    $6D      MASTER INITIALIZATION FLAG
                         *              (IF EQUAL "1" - PERFORM MASTER INITIALIZATION
          006E  A MRAM   EQU    $6E      RAM DESCRIPTION FLAG (2 BYTES)
                         *               SET TO MAX ADDRESS OF RAM AVAILABLE (MINUS STACK AREA) EG.
                         *               = $03DF - 1KB RAM
                         *               = $07DF - 2KB RAM
                         *               = $4FDF - 20KB RAM
```

```
                   *
                   *
                   *      PAGE 0 COLUMN 7   DATA POINTERS
                   *
                   *
0070  A VOLDAY EQU    $70      NUMBER OF DAYS VOLUME DATA COLLECTED
0071  A VODAY1 EQU    $71      POINTER TO BEGINNIG OF THE VOLUME BUFFER - 2 BYTES
0073  A VOBUF1 EQU    $73      LENGTH OF FIRST DAY VOLUME BUFFER (OTHER DAYS THE SAME) - 2 BYTES
        *
0075  A SPDDAY EQU    $75      NUMBERS OF DAYS SPEED DATA COLLECTED  (MAX 5)
0076  A SPDAY1 EQU    $76      POINTER TO THE BEGINNING OF THE SPEED BUFFER - 2 BYTES
0078  A SPBUF  EQU    $78      LENGTH OF SPEED BUFFER (PER DAY) - 2 BYTES
        *
007A  A LENDAY EQU    $7A      NUMBER OF DAYS LENGTH DATA COLLECTED (MAX 5)
007B  A LNDAY1 EQU    $7B      POINTER TO BEGINNING OF THE LENGTH BUFFER - 2 BYTES
007D  A LNBUF1 EQU    $7D      LENGTH OF LENGTH BUFFER (PER DAY) - 2 BYTES
        *
007F  A DAYMAX EQU    $7F      MAX NUMBER OF DAYS OF DATA POSSIBLE WITH GIVEN MEMORY SIZE
        *                      -CURRENTLY SET TO 5
```

```
PAGE  010  EQUATE  .SA:1

                        *
                        *
                        *          PAGE 0 COLUMN 8    OTHER VARIABLES
                        *
                        *          THESE LOCATIONS ARE USED DURING PROM CHECKSUM CALCULATIONS AND THE RAM TEST
                        *
                 0080  A TEMP03 EQU    $80       TEMP LOCATION USED BY CALPRO ROUTINE AND POWRON ROUTINE
                 0081  A TEMP04 EQU    $81       TEMP LOCATION USED BY CALPRO ROUTINE AND POWRON ROUTINE
                 0082  A PCHECK EQU    $82       PROM CHECKSUM SAVE AREA - 2 BYTES (OR RAM CHIP BIT INDICATOR FOR RAM TEST)
                 0084  A PADDR  EQU    $84       PROM CHECKSUM OR RAM TEST ADDRESS SAVE AREA - 2 BYTES
                 0086  A PRERFL EQU    $86       PROM ERROR FLAG
                         *                  BIT 0 - PROM C000 TO C7FF
                         *                  BIT 1 - PROM C800 TO CFFF
                         *                  BIT 2 - PROM D000 TO D7FF
                         *                  BIT 3 - PROM D800 TO DFFF
                         *                  BIT 4 - PROM E000 TO E7FF
                         *                  BIT 5 - PROM E800 TO EFFF
                         *                  BIT 6 - PROM F000 TO F7FF
                         *                  BIT 7 - PROM F800 TO FFFF
                         *
                         *         TIME AND DATE OF LAST POWER FAILURE
                         *
                 0087  A LFDATE EQU    $87       DATE OF LAST POWER FAILURE - 3 BYTES - YEAR, MONTH, DAY (BCD)
                 0087  A LFYEAR EQU    $87       YEAR OF LAST POWER FAILURE
                 0088  A LFMNTH EQU    $88       MONTH OF LAST POWER FAILURE
                 0089  A LFDAY  EQU    $89       DAY OF LAST POWER FAILURE
                 008A  A LFTIME EQU    $8A       TIME OF LAST POWER FAILURE - 2 BYTES - HOURS AND MINUTES (BCD)
                 008A  A LFHR   EQU    $8A       HOUR OF LAST POWER FAILURE
                 008B  A LFMIN  EQU    $8B       MINUTE OF LAST POWER FAILURE
                         *
                         *         DETECTOR DESCRIPTORS
                         *
                 008C  A DETER1 EQU    $8C       DETECTOR ERROR WORD - DIRECTION 1, UPSTREAM DETECTORS
                 008D  A DETER2 EQU    $8D       DETECTOR ERROR WORD - DIRECTION 1, DOWNSTREAM DETECTORS
                 008E  A DETER3 EQU    $8E       DETECTOR ERROR WORD - DIRECTION 2, UPSTREAM DETECTORS
                 008F  A DETER4 EQU    $8F       DETECTOR ERROR WORD - DIRECTION 2, DOWNSTREAM DETECTORS
                         *                       BIT 0 = LANE 1 (LEFTMOST IE. CENTRE LANE)
                         *                       BIT 7 = LANE 8 (RIGHTMOST IE. SHOULDER LANE)
```

```
PAGE  011  EQUATE  .SA:1

                       *
                       *
                       *         LAST PAGE OF RAM COLUMNS E AND F
                       *
                       *       0 STACK AREA - STACK REQUIRES 32 BYTES
                       *
           03FF  A STAK1  EQU    $3FF      STACK POINTER - 1K RAM
           07FF  A STAK2  EQU    $7FF      STACK POINTER - 2K RAM
           0008  A TWOK   EQU    $08       2K INCREMENT = $0800
           4FDF  A EOFRAM EQU    $4FDF     END OF USEFUL RAM - 20K MEMORY
```

```
PAGE  012  EQUATE  .SA:1

                         *
                         *
                         *        INPUT/OUTPUT PORTS
                         *
                         *
             5000   A PORT0  EQU    $5000    DOWNTIME ACC. - MINUTES (0 TO 255)
                         *
             5007   A PORT7  EQU    $5007    KEYBOARD ENTRY  (INPUT)
                         *
                         *                      BIT
                         *
                         *                       1       KEYBOARD CONTROL-VALID KEYPUSH IF SET
                         *                       2       KEYBOARD CHARACTER-LEAST SIG. BIT
                         *                       3       KEYBOARD CHARACTER-NEXT " "
                         *                       4       KEYBOARD CHARACTER-NEXT MOST SIG. BIT
                         *                       5       KEYBOARD CHARACTER-MOST SIG. BIT
                         *                       6       STOP TIMING
                         *                       7       NOT CONNECTED
                         *                       8       NOT CONNECTED
                         *
             5008   A PORT8  EQU    $5008    DISPLAY CONTROL PORT  (OUTPUT) - FRONT PANEL DISPLAY
                         *
                         *    "CC" - CHARACTER CONTROL OR BLANKING
                         *
                         *                      BIT
                         *
                         *                       1       CC-PHASE
                         *                       2       CC-INTERVAL
                         *                       3       CC-TIMING LS
                         *                       4       CC-TIMING NLS
                         *                       5       CC-TIMING NMS
                         *                       6       CC-TIMING MS
                         *                       7       CALL LT 8
                         *                       8       CALL LT 9
                         *
                         *
             5009   A PORT9  EQU    $5009    DISPLAY CHAR. DATA AND BLANKING  (OUTPUT) - FRONT PANEL DISPLAY
                         *
                         *                     BIT
                         *
                         *                      1          CHARACTER LEAST SIG.
                         *                      2          CHARACTER NEXT LEAST SIG.
                         *                      3          CHARACTER NEXT MOST SIG.
                         *                      4          CHARACTER MOST SIG.
                         *                      5          DECIMAL POINT
                         *                      6          BLANKING - PHASE AND INTERVAL
                         *                      7          BLANKING - TIMING
                         *                      8          NOT CONNECTED
                         *
             500A   A PORTA  EQU    $500A    BITS 0-7 = CALL LITES 0-7
             500F   A PORTF  EQU    $500F    DOWNTIME ACCUMULATOR - SECONDS (0 TO 59)
                         *
             5FFF   A RTCLOK EQU    $5FFF    REAL TIMU CLOCK RESETTING LOCATION
                         *
             6000   A ACIACS EQU    $6000    ACIA CONTROL/STATUS REG
             6001   A ACIADA EQU    $6001    ACIA DATA REGISTER
```

```
                   *
                   *
                   *      EPROM  CHIPS
                   *
                   *       SUBROUTINE ADDRESSES - FREQUENCY OF EXUCUTION IS GOVERNED BY TABLES IN EXEC ROUTINE
                   *
                   *
                   *       APPLICATION PROGRAMS
                   *
                   *           CHIP # 1        FROM    $C000
                   *
        C350  A PCS2    EQU    $C350
        C400  A PCS4    EQU    $C400
        C4A0  A PCS5    EQU    $C4A0
        C550  A PCS6    EQU    $C550
        C6A0  A PCS7    EQU    $C6A0
                *
                *              CHIP # 2        FROM    $C800
                *
        C840  A PCS4A   EQU    $C840
        C8D0  A PCS5A   EQU    $C8D0
        C950  A PCS6A   EQU    $C950
        CBB0  A PCS3B   EQU    $CBB0
        CC00  A PCS3C   EQU    $CC00
        CC50  A PCS8    EQU    $CC50
        CDA0  A PCS8A   EQU    $CDA0
        CF00  A PCS7A   EQU    $CF00
                *
                *              CHIP # 3        FROM    $D000
                *
        D000  A INIT    EQU    $D000
        D250  A HEADER  EQU    $D250
        D4A0  A PCS3    EQU    $D4A0
        D550  A PCS3A   EQU    $D550
                *
                *              CHIP # 4        FROM    $D800
                *
                *       EMPTY
                *
                *
                *              CHIP # 5        FROM    $E000
                *
                *       EMPTY
                *
                *
                *              CHIP # 6        FROM    $E800
                *
                *       EMPTY
                *
                *
```

```
PAGE  014  EQUATE  .SA:1

00455                   *
00456                   *         SYSTEMS PROGRAMS
00457                   *
00458                   *              CHIP # 7       FROM   $F000
00459                   *
00460           F000  A COMM   EQU     $F000
00461           F12F  A TFIRST EQU     $F12F
00462           F240  A PREANS EQU     $F240
00463           F670  A TIMER1 EQU     $F670
00464           F6C0  A TIMER2 EQU     $F6C0
00465           F6F0  A MASINT EQU     $F6F0
00466           F750  A RAM60  EQU     $F750
00467           F760  A RAMCHK EQU     $F760
00468                   *
00469                   *              CHIP # 8      FROM   $F800
00470                   *
00471           F800  A KEYBD  EQU     $F800
00472           FB00  A ITEST  EQU     $FB00
00473           FB0A  A CLOCK  EQU     $FB0A
00474           FB96  A SWI    EQU     $FB96
00475           FCD0  A POWRON EQU     $FCD0
00476           FE30  A EXEC   EQU     $FE30
00477           FF00  A CHEK24 EQU     $FF00
00478           FF20  A CHEK60 EQU     $FF20
00479           FF30  A CALPRO EQU     $FF30
00480           FFF6  A IGNORE EQU     $FFF6
00481           FFF8  A VECTOR EQU     $FFF8
```

```
PAGE  015  EQUATE  .SA:1

00483                      *
00484                      *
00485                      *         SYMBOLS USED IN ASSEMBLY
00486                      *
00487                      *
00488          0001  A TBIT0   EQU     $01       BIT 0
00489          0002  A TBIT1   EQU     $02       BIT 1
00490          0004  A TBIT2   EQU     $04       BIT 2
00491          0008  A TBIT3   EQU     $08       BIT 3
00492          0010  A TBIT4   EQU     $10       BIT 4
00493          0020  A TBIT5   EQU     $20       BIT 5
00494          0040  A TBIT6   EQU     $40       BIT 6
00495          0080  A TBIT7   EQU     $80       BIT 7
```

```
PAGE  001  EQUATE  .SA:1  COMM

       OPT    L
       NAM    COMM
*
*        COMMUNICATIONS SUBROUTINE FOR PCS SYSTEM
*
*
*             ****************************************************
*             *                                                  *
*             *   PROGRAMMER:   W.J. WIERCIENSKI                 *
*             *                                                  *
*             *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS  *
*             *   TRANSPORTATION TECHNOLOGY AND ENERGY           *
*             *   1201 WILSON AVENUE                             *
*             *   DOWNSVIEW, ONTARIO  M3M 1J8                    *
*             *                                                  *
*             *   TELEPHONE (416) 248-7141                       *
*             *                                                  *
*             *   NOVEMBER  1982 AD                              *
*             *                                                  *
*             ****************************************************
*
*
*        THE COMMUNICATIONS ROUTINE IS ENVOKED BY THE ACIA INTERRUPT
*
*        NORMALLY ACIA IS IN THE RECEIVING MODE. THE NOVA INITIATES THE MESSAGE,
*        THE FIELD MODEM PERFORMS THE HAND-SHAKING OPERATION AND ONLY AFTER
*        THE FIRST CHARACTER HAS BEEN RECEIVED IS THE ACIA AWARE THAT THE LINE IS OPEN.
*
*        THE CONVERSATION IS SERIAL, IE. THE MESSAGE IS RECEIVED, THE RECEIVER
*        IS TURNED OFF, THE TRANSMITTER IS TURNED ON AND THE ANSWER IS
*        TRANSMITTED BACK. AT THE END OF THE CONVERSATION THE RECEIVER IS TURNED ON AGAIN.
*
*
*        AS THE COMMUNICATIONS ROUTINE IS INTERRUPT DRIVEN IT MAY NOT REALIZE THAT
*        THE NOVA COMPUTER IS INACTIVE OR THAT THE LINE WENT DOWN. CONSEQUENTLY IN
*        ORDER TO BREAK THE DEADLOCK 2 TIMERS ARE EMPLOYED. THESE ARE EXAMINED PERIODICALLY.
*        IF THE DEADLOCK IS DETECTED WHILE RECEIVING,THE APPROPRIATE MESSAGE IS SENT BACK.
*        IF THERE WAS A DEADLOCK DURING TRANSMISSION ( DUE TO THE DATA CARRIER LOSS )
*        THE CONTROLLER WILL SWITCH BACK TO RECEIVING.
*
*
*
*        THE MESSAGES ARE ASSEMBLED BY THE RECEIVING ROUTINE RECV, PROCESSED BY THE PREPARE
*        ANSWER ROUTINE PREANS, AND THE ANSWER IS TRANSMITTED BACK BY THE TRANSMITTING ROUTINE
*        XMIT
*        RECEIVING ROUTINE RECV EXAMINES THE MESSAGE INITIALLY AND EITHER PASSES IT FOR
*        SUBSEQUENT PROCESSING TO PREPARE ANSWER SUBROUTINE PREANS OR RESPONDS WITH A
*        NEGATIVE ACKNOWLEDGEMENT MESSAGE ( CODE 010 ). THE FOLLOWING CONDITIONS WILL
*        PRECIPITATE THIS MESSAGE
*        - INPUT ERROR ( PARITY, FRAMING, OR OVER RUN )
*        - INCORRECT CONTROLLER ADDRESS
*        - INCORRECT LENGTH OF MESSAGE
*        - INCORRECT CHECKSUM
*        IF THE FORMAT OF THE MESSAGE IS CORRECT FROM THE POINT OF VIEW OF THE RECEIVING
*        SUBROUTINE IT IS PASSED FOR PROCESSING TO THE PREPARE ANSWER SUBROUTINE PREANS.
*        THIS SUBROUTINE MAY GENERATE EITHER THE PROPER ANSWER OR A LOGICAL ERROR MESSAGE
*        ( CODE 001 ) DEPENDING ON WHETHER THE REQUEST IS LOGICAL AND COMPLETE.
*        THE CONDITIONS WHICH CAN CAUSE THE LOGICAL ERROR ANSWER ARE LISTED AS APLLICABLE
```

```
PAGE  002  COMM     .SA:1  COMM

00556                 *       FOR VARIOUS MESSAGES IN THE PREPARE ANSWER SUBROUTINE.
00557                 *
00558                 *
```

```
PAGE  003  COMM    .SA:1  COMM

00560A F000                     ORG    COMM
00561                   *
00562                   *         MAIN PROGRAM
00563                   *
00564                   *
00565                   *         ON ENTRY REGISTER A CONTAINS THE VALUE FROM THE ACIA CONTROL/STATUS REGISTER.
00566                   *
00567                   *
00568A F000 44                  LSRA             GET RECEIVER STATUS
00569A F001 24 04 F007          BCC    P1        READER IS NOT FULL - CHECK TRANSMITTER
00570A F003 BD F023  A          JSR    RECV      READER IS FULL, GO TO RECEIVE ROUTINE
00571A F006 3B                  RTI
00572A F007 44          P1      LSRA             GET TRANSMITTER STATUS
00573A F008 24 04 F00E          BCC    P2        TRANSMITTER DATA REGISTER IS NOT EMPTY - CHECK FURTHER
00574A F00A BD F152  A          JSR    XMIT      TRANSMITTER DATA REGISTER IS EMPTY - GO TO TRANSMIT ROUTINE
00575A F00D 3B                  RTI
00576                   *
00577A F00E F6 6001  A  P2      LDAB   ACIADA    READ DATA REGISTER TO CLEAR INTERRUPT
00578A F011 44                  LSRA             GET DATA CARRIER DETECT STATUS
00579A F012 25 01 F015          BCS    P3        BRANCH IF DATA CARRIER LOST
00580A F014 3B                  RTI
00581                   *
00582                   *         DATA CARRIER LOST - CHECK FURTHER
00583                   *
00584A F015 D6 45    A  P3      LDAB   RBITS     GET RECEIVER INDICATOR
00585A F017 C1 01    A          CMPB   #1        CHECK IF NORMAL CARRIER LOSS
00586A F019 27 07 F022          BEQ    P4        YES
00587                   *
00588                   *         CARRIER LOSS IN THE MIDDLE OF THE MESSAGE
00589                   *
00590A F01B C6 01    A          LDAB   #1
00591A F01D D7 45    A          STAB   RBITS     PREPARE FOR
00592A F01F 7F 0046  A          CLR    XBITS     A NEW MESSAGE
00593A F022 3B          P4      RTI
```

```
PAGE  004  COMM    .SA:1  COMM

00595                    *
00596                    *
00597                    *        RECEIVING SUBROUTINE
00598                    *
00599                    *        PROCESSES CHARACTERS RECEIVED AFTER CHECKING FOR INPUT ERRORS,
00600                    *        PREPARES ANSWER & STARTS TRANSMISSION WHEN END OF MSG IS DETECTED.
00601                    *
00602                    *        STRUCTURE OF THE INPUT MESSAGE IS AS FOLLOWS
00603                    *
00604                    *        BYTE #         FUNCTION
00605                    *          1              TRANSMISSION CODE/ADDRESS
00606                    *                            BITS 0 TO 4 - STATION ADDRESS
00607                    *                            BITS 5 TO 7 - MESSAGE CODE
00608                    *          2             BYTE COUNT TO CHECKSUM
00609                    *                        LENGTH OF MESSAGE FROM TRANSMISSION CODE/ADDRESS BYTE TO CHECKSUM BYTE
00610                    *                        INCLUSIVE.
00611                    *          3 TO X        BODY OF MESSAGE, VARIOUS LENGTHS (3 TO 19 BYTES) DEPENDENT ON THE
00612                    *                        MESSAGE TYPE.
00613                    *                        - FURTHER DESCRIBED FOR EVERY MESSAGE IN THE PREANS SUBROUTINE
00614                    *          X+1           CHECKSUM CALCULATED USING THE EXCLUSIVE OR OPERATION ( STARTING
00615                    *                        FROM FIRST BYTE THROUGH TO BYTE X )
00616                    *          X+2           FIRST END CHARACTER BYTE           =$AA
00617                    *          X+3           SECOND END CHARACTER BYTE          =$55
00618                    *          X+4           THIRD END CHARACTER BYTE           =$55
00619                    *
00620                    *
00621A F023 F6 6001  A RECV   LDAB   ACIADA   READ CHARACTER RECEIVED
00622A F026 84 38    A        ANDA   #%00111000 CHECKS FOR INPUT ERRORS (PARITY, OVERRUN, OR FRAMING)
00623A F028 27 16 F040        BEQ    R1       NO ERRORS
00624                    *
00625                    *      INPUT ERROR
00626                    *
00627A F02A 96 45    A        LDAA   RBITS    GET RBITS
00628A F02C 81 01    A        CMPA   #1       IS THIS THE FIRST BYTE OF MESSAGE
00629A F02E 26 05 F035        BNE    RE1      NO
00630A F030 7F 0063  A        CLR    CTIME1   YES - START TIMER
00631A F033 20 07 F03C        BRA    RE2
00632A F035 7C 004F  A RE1    INC    CERROR   INCREMENT COMMUNICATION ERROR COUNT
00633A F038 81 07    A        CMPA   #7       CHECK FOR PREVIOUS ERRORS
00634A F03A 22 03 F03F        BHI    RE3      YES, NO NZED TO SET UP FOR END OF MESSAGE SEARCH
00635A F03C BD F110  A RE2    JSR    EOMAF    PREPARE FOR END OF MESSAGE AFTER FAILURE
00636A F03F 39         RE3    RTS             EXIT
00637                    *
00638                    *      NO HARDWARE INPUT ERRORS
00639                    *
00640A F040 96 45    A R1     LDAA   RBITS    RBITS TO REG A - TO CHECK WHERE WE ARE
00641A F042 81 01    A        CMPA   #1       CHECK FOR BEGINNING OF MESSAGE
00642A F044 26 1C F062        BNE    R2       NO
00643                    *
00644                    *      RBITS=1 - BEGINNING OF MESSAGE BYTE EXPECTED
00645                    *
00646A F046 7F 0063  A        CLR    CTIME1   START TIMER
00647A F049 D7 40    A        STAB   TCODE    SAVE MESSAGE BYTE
00648A F04B C4 1F    A        ANDB   #%00011111 DISCARD MESSAGE CODE
00649A F04D D1 00    A        CMPB   PCSCAD   CHECK FIELD STATION ADDRESS
00650A F04F 27 04 F055        BEQ    R11      OK - PROCESS THIS BYTE
00651A F051 BD F110  A        JSR    EOMAF    NOT CORRECT - EXIT
00652A F054 39                RTS             EXIT
```

```
PAGE  005  COMM     .SA:1  COMM

00653A F055 D6 40     A R11     LDAB   TCODE    RESTORE MESSAGE BYTE
00654A F057 D7 43     A         STAB   CHECKS   START CHECKSUM
00655A F059 C4 E0     A         ANDB   #%11100000 ELIMINATE FIELD STATION ADDRESS
00656A F05B D7 40     A         STAB   TCODE    SAVE TRANSACTION CODE
00657A F05D 86 02     A         LDAA   #2       RBITS=2
00658A F05F 97 45     A         STAA   RBITS    SAVE IT
00659A F061 39                  RTS             EXIT
00660                 *
00661                 *           RBITS NOT 1 - CHECK FOR 2
00662                 *
00663A F062 81 02     A R2      CMPA   #2       CHECK FOR ' # OF BYTES TO CHECKSUM '
00664A F064 26 1F F085          BNE    R3       NO - CHECK AGAIN
00665                 *
00666                 *           RBITS=2 - 'NUMBER OF BYTES TO CHECKSUM' EXPECTED
00667                 *
00668A F066 C1 03     A         CMPB   #3       CHECK FOR MIN VALUE OF LENGTH
00669A F068 23 04 F06E          BLS    R221     LENGTH TO SHORT - EXIT
00670A F06A C1 16     A         CMPB   #22      CHECK FOR MAX. LENGTH
00671A F06C 23 04 F072          BLS    R22      OK - PROCESS IT
00672A F06E BD F110   A R221    JSR    EOMAF    LENGTH TO LONG - EXIT
00673A F071 39                  RTS             EXIT
00674A F072 17          R22     TBA             SAVE LENGTH IN REGISTER A
00675A F073 D8 43     A         EORB   CHECKS   CALCULATES NEW CHECKSUM
00676A F075 D7 43     A         STAB   CHECKS   SAVE NEW CHECKSUM
00677A F077 80 03     A         SUBA   #3       CORRECT FOR LENGTH OF MESSAGE BODY
00678A F079 97 49     A         STAA   DLNGTH   SAVE IT
00679A F07B CE 0050   A         LDX    #INDATA  GET BEGINNING OF INPUT BUFFER
00680A F07E DF 47     A         STX    DPOINT   PREPARE POINTER
00681A F080 86 03     A         LDAA   #3       RBITS=3
00682A F082 97 45     A         STAA   RBITS    SAVE IT
00683A F084 39                  RTS             EXIT
00684                 *
00685                 *           RBITS NOT 2 - CHECK FOR 3
00686                 *
00687A F085 81 03     A R3      CMPA   #3       CHECK FOR TEXT BYTE
00688A F087 26 18 F0A1          BNE    R4       NO - CHECK AGAIN
00689                 *
00690                 *           RBITS=3 - MESSAGE TEXT BYTE EXPECTED
00691                 *
00692A F089 DE 47     A         LDX    DPOINT   GET BUFFER POINTER
00693A F08B E7 00     A         STAB   0,X      STORE INTO BUFFER
00694A F08D D8 43     A         EORB   CHECKS   CALCULATE NEW CHECKSUM
00695A F08F D7 43     A         STAB   CHECKS   SAVE IT
00696A F091 96 49     A         LDAA   DLNGTH   GET LENGTH
00697A F093 4A                  DECA            SUBTRACT 1 FOR BYTE JUST OBTAINED
00698A F094 97 49     A         STAA   DLNGTH   SAVE NEW LENGTH
00699A F096 26 05 F09D          BNE    R31      CHECK FOR LAST TEXT BYTE
00700A F098 86 04     A         LDAA   #4       LAST TEXT BYTE OBTAINED, RBITS=4
00701A F09A 97 45     A         STAA   RBITS    SAVE IT
00702A F09C 39                  RTS             EXIT
00703A F09D 08          R31     INX             POINT TO THE NEXT EMPTY SPACE IN THE INPUT BUFFER
00704A F09E DF 47     A         STX    DPOINT   SAVE IT
00705A F0A0 39                  RTS             EXIT
00706                 *
00707                 *           RBITS NOT 3 - CHECK FOR 4
00708                 *
00709A F0A1 81 04     A R4      CMPA   #4       CHECK FOR CHECKSUM BYTE
00710A F0A3 26 10 F0B5          BNE    R5       NO - CHECK AGAIN
```

```
PAGE  006  COMM     .SA:1  COMM

00711                       *
00712                       *    RBITS=4 - CHECKSUM BYTE EXPECTED
00713                       *
00714A F0A5 D1 43     A          CMPB   CHECKS    COMPARE CHECKSUMS
00715A F0A7 27 04 F0AD           BEQ    R41       OK - PREPARE ANSWER
00716A F0A9 BD F110   A          JSR    EOMAF     CHECKSUMS NOT EQUAL - EXIT
00717A F0AC 39                   RTS              EXIT
00718                       *
00719                       *    CHECKSUMS AGREE - LOGICAL MESSAGE RECEIVED
00720                       *
00721A F0AD BD F240   A R41      JSR    PREANS    PROCESS LOGICAL MESSAGE
00722A F0B0 86 05     A          LDAA   #5        RBITS=5
00723A F0B2 97 45     A          STAA   RBITS     SAVE IT
00724A F0B4 39                   RTS              EXIT
00725                       *
00726                       *    RBITS NOT 4 - CHECK FOR 5
00727                       *
00728A F0B5 81 05     A R5       CMPA   #5        CHECK FOR FIRST END CHARACTER
00729A F0B7 26 0C F0C5           BNE    R6        NO - CHECK AGAIN
00730                       *
00731                       *    RBITS=5 - 1-ST END CHARACTER EXPECTED
00732                       *
00733A F0B9 C1 AA     A          CMPB   #%10101010 IS VALUE CORRECT?
00734A F0BB 27 03 F0C0           BEQ    R51       YES
00735A F0BD 7C 004F   A          INC    CERROR    NO - INCREMENT ERROR COUNT
00736A F0C0 86 06     A R51      LDAA   #6        RBITS=6
00737A F0C2 97 45     A          STAA   RBITS     SAVE IT
00738A F0C4 39                   RTS              EXIT
00739                       *
00740                       *    RBITS NOT 5 - CHECK FOR 6
00741                       *
00742A F0C5 81 06     A R6       CMPA   #6        CHECK FOR SECOND END CHARACTER
00743A F0C7 26 0C F0D5           BNE    R7        NO - CHECK AGAIN
00744                       *
00745                       *    RBITS=6 - 2-ND END CHARACTER EXPECTED
00746                       *
00747A F0C9 C1 55     A          CMPB   #%01010101 IS VALUE CORRECT?
00748A F0CB 27 03 F0D0           BEQ    R61       YES
00749A F0CD 7C 004F   A          INC    CERROR    NO - INCREMENT ERROR COUNT
00750A F0D0 86 07     A R61      LDAA   #7        RBITS=7
00751A F0D2 97 45     A          STAA   RBITS     SAVE IT
00752A F0D4 39                   RTS              EXIT
00753                       *
00754                       *    RBITS NOT 6 - CHECK FOR 7
00755                       *
00756A F0D5 81 07     A R7       CMPA   #7        CHECK FOR THIRD END CHARACTER
00757A F0D7 26 0B F0E4           BNE    R8        NO - CHECK AGAIN
00758                       *
00759                       *    RBITS=7 - 3-RD END CHARACTER EXPECTED
00760                       *
00761A F0D9 C1 55     A          CMPB   #%01010101 IS VALUE CORRECT?
00762A F0DB 27 03 F0E0           BEQ    R71       YES
00763A F0DD 7C 004F   A          INC    CERROR    NO - INCREMENT ERROR COUNT
00764A F0E0 BD F12F   A R71      JSR    TFIRST    TRANSMIT FIRST CHARACTER OF PREPARED RESPONSE TO REQUEST
00765A F0E3 39                   RTS              EXIT
00766                       *
00767                       *    THIS PART OF RECEIVING S/R DEALS WITH THE ERROR SITUATION
00768                       *
```

```
PAGE  007  COMM    .SA:1  COMM

00769                      *
00770                      *      RBITS NOT 7 - CHECK FOR 8
00771                      *
00772A F0E4 81 08     A R8     CMPA   #8        CHECK  FOR FIRST END CHARACTER
00773A F0E6 26 09 F0F1         BNE    R9        NO -CHECK AGAIN
00774                      *
00775                      *      RBITS=8 - 1-ST END CHARACTER AFTER FAILURE EXPECTED
00776                      *
00777A F0E8 C1 AA     A        CMPB   #%10101010 IS VALUE FOUND?
00778A F0EA 26 04 F0F0         BNE    R81       NO
00779A F0EC 86 09     A        LDAA   #9        YES, RBITS=9
00780A F0EE 97 45     A        STAA   RBITS     SAVE IT
00781A F0F0 39          R81    RTS              EXIT
00782                      *
00783                      *      RBITS NOT 8 - CHECK FOR 9
00784                      *
00785A F0F1 81 09     A R9     CMPA   #9        CHECK FOR SECOND END CHARACTER
00786A F0F3 26 0E F103         BNE    R10       NO - RBITS MUST EQUAL 10
00787                      *
00788                      *      RBITS=9 - 2-ND END CHARACTER AFTER FAILURE EXPECTED
00789                      *
00790A F0F5 C1 55     A        CMPB   #%01010101 IS VALUE FOUND?
00791A F0F7 26 05 F0FE         BNE    R91       NO
00792A F0F9 86 0A     A        LDAA   #10       RBITS=10 - YES, SEARCH FOR LAST END CHARACTER
00793A F0FB 97 45     A        STAA   RBITS     SAVE IT
00794A F0FD 39                 RTS              EXIT
00795A F0FE 86 08     A R91    LDAA   #8        RBITS=8 NO, TRY AGAIN
00796A F100 97 45     A        STAA   RBITS     SAVE IT
00797A F102 39                 RTS              EXIT
00798                      *
00799                      *      RBITS=10 - 3-ND END CHARACTER AFTER FAILURE EXPECTED
00800                      *
00801A F103 C1 55     A R10    CMPB   #%01010101 IS VALUE FOUND?
00802A F105 26 04 F10B         BNE    R101      NO - START SEARCH AGAIN
00803A F107 BD F12F   A        JSR    TFIRST    YES - START TRANSMISSION
00804A F10A 39                 RTS              EXIT
00805A F10B 86 08     A R101   LDAA   #8        RBITS=8 - START SEARCH AGAIN
00806A F10D 97 45     A        STAA   RBITS     SAVE IT
00807A F10F 39                 RTS              EXIT
00808                      *
00809                      *      END OF MAIN RECEIVING SUBROUTINE
00810                      *
```

```
PAGE  008  COMM    .SA:1  COMM

00812                        *
00813                        *
00814                        *        PREPARE FOR END OF MESSAGE AFTER FAILURE SUBROUTINE
00815A F110 86 08     A  EOMAF  LDAA  #8          PUT VALUE INTO ACCUMULATOR
00816A F112 97 45     A         STAA  RBITS       SET RBITS=8
00817A F114 96 44     A         LDAA  ERCODE      GET ERROR CODE
00818A F116 8A 08     A         ORAA  #%00001000 SET COMMUNICATIONS ERROR BIT
00819A F118 97 44     A         STAA  ERCODE      STORE BACK TO ERROR CODE BYTE
00820                        *
00821                        *
00822                        *        PREPARE ERROR MESSAGE
00823A F11A 96 00     A         LDAA  PCSCAD      GET FIELD STATION ADDRESSS
00824A F11C 8B 40     A         ADDA  #%01000000 INTRODUCE MESSAGE CODE
00825A F11E 97 40     A         STAA  TCODE       SAVE IT
00826A F120 7F 0041   A         CLR   MSGLEN      CLEAR MSG OF MESSAGE LENGTH
00827A F123 86 05     A         LDAA  #5          ASSEMBLE LSB OF MESSAGE LENGTH
00828A F125 97 42     A         STAA  MSGLEN+1    STORE IT
00829A F127 CE 0000   A         LDX   #0          CLEAR REG X
00830A F12A DF 47     A         STX   DPOINT      CLEAR POINTER TO BODY OF MSG
00831A F12C DF 49     A         STX   DLNGTH      NO BODY OF MSG
00832A F12E 39                  RTS               EXIT
```

```
PAGE  009  COMM    .SA:1  COMM

00834                    *
00835                    *
00836                    *   TRANSMIT 1-ST CHARACTER SUBROUTINE
00837                    *
00838                    *   THIS SUBROUTINE TURNS THE RECEIVER OFF & THE TRANSMITTER ON AND
00839                    *   THEN SENDS THE 1-ST CHARACTER OF THE MESSAGE.
00840                    *   IT IS INVOKED EITHER BY THE END OF THE MESSAGE OR BY THE EXPIRATION
00841                    *   OF THE RECEIVING TIMER.
00842                    *
00843                    *
00844A F12F                 ORG   TFIRST
00845A F12F B6 6000  A      LDAA  ACIACS    GET ACIA STATUS REGISTER
00846A F132 84 04    A      ANDA  #%00000100 CHECK FOR  DATA CARRIER PRESENCE
00847A F134 27 05 F13B      BEQ   TF1       OK
00848A F136 86 01    A      LDAA  #1        LINE DOWN - PUT 1 INTO ACCUMULATOR
00849A F138 97 45    A      STAA  RBITS     AND GO BACK TO RECEIVING (RBITS=1)
00850A F13A 39              RTS             EXIT
00851                    *
00852                    *   LINE OPEN
00853                    *
00854A F13B 86 3A    A TF1  LBAA  #%00111010 RECEIVER OFF, TRANSMITTER ON
00855                    *                  (8 DATA BITS, EVEN PARITY, 1 STOP BIT, 300 BAUD)
00856A F13D B7 6000  A      STAA  ACIACS    DO IT
00857A F140 7F 0045  A      CLR   RBITS     NO RECEIVING NOW (RBITS=0)
00858A F143 96 40    A      LDAA  TCODE     GET 1-ST BYTE OF MESSAGE
00859A F145 B7 6001  A      STAA  ACIADA    SEND IT
00860A F148 97 43    A      STAA  CHECKS    START CHECKSUM
00861A F14A 86 01    A      LDAA  #1        PREPARE FOR NEXT BYTE (XBITS=1)
00862A F14C 97 46    A      STAA  XBITS     SAVE IT
00863A F14E 7F 0064  A      CLR   CTIME2    START TIMER #2
00864A F151 39              RTS             EXIT
```

```
PAGE  010  COMM    .SA:1  COMM

00866                    *
00867                    *
00868                    *        TRANSMITTING ROUTINE
00869                    *
00870                    *        IT IS ENTERED VIA TRANSMIT INTERRUPTS AFTER FIRST (OR SUBSEQUENT)
00871                    *        CHARACTERS OF THE MESSAGE HAVE LEFT THE ACIA'S BUFFER.
00872                    *
00873                    *        IT EITHER SENDS THE NEXT BYTE OF THE MESSAGE OR SWITCHES BACK TO
00874                    *        RECEIVING AS APPLICABLE.
00875                    *
00876                    *        STRUCTURE OF THE OUTPUT MESSAGE IS AS FOLLOWS
00877                    *
00878                    *          BYTE #         FUNCTION
00879                    *            1              TRANSMISSION CODE/ADDRESS
00880                    *                              BITS 0 TO 4 - STATION ADDRESS
00881                    *                              BITS 5 TO 7 - MESSAGE CODE
00882                    *            2&3            BYTE COUNT TO CHECKSUM
00883                    *                           LENGTH OF MESSAGE FROM TRANSMISSION CODE/ADDRESS BYTE TO CHECKSUM
00884                    *                           BYTE INCLUSIVE
00885                    *            4 TO X         BODY OF MESSAGE, VARIOUS LENGTHS ( FROM 0 TO 431 BYTES ),
00886                    *                           DEPENDENT ON MESSAGE TYPE.
00887                    *                           - FURTHER DESCRIBED FOR EVERY MESSAGE IN PREANS SUBROUTINE
00888                    *            X+1            ERROR CODE BYTE
00889                    *            X+2            CHECKSUM - CALCULATED USING THE EXCLUSIVE OR OPERATION
00890                    *                           ( STARTING FROM THE FIRST BYTE THROUGH TO BYTE X+1 ).
00891                    *            X+3            FIRST END CHARACTER BYTE          =$AA
00892                    *            X+4            SECOND END CHARACTER BYTE         =$55
00893                    *            X+5            THIRD END CHARACTER BYTE          =$55
00894                    *            X+6            FOURTH END CHARACTER BYTE         =$55
00895                    *
00896                    *
00897A F152 96 46    A XMIT   LDAA   XBITS    GET TRANSMIT INDICATOR
00898A F154 81 0A    A        CMPA   #10      CHECK FOR LAST INTERRUPT
00899A F156 27 07 F15F        BEQ    X0       BRANCH IF LAST CHARACTER SENT
00900A F158 F6 6000  A        LDAB   ACIACS   GET STATUS REGISTER
00901A F15B C4 04    A        ANDB   #%00000100 CHECK DATA CARRIER LOSS BIT
00902A F15D 27 0D F16C        BEQ    X1       OK - LINE OPEN
00903A F15F 7F 0046  A X0     CLR    XBITS    LINE DOWN
00904A F162 C6 9A    A        LDAB   #%10011010 RECEIVER ON, TRANSMITTER OFF
00905                    *                    (8 DATA BITS, EVEN PARITY, 1 STOP BIT, 300 BAUD)
00906A F164 F7 6000  A        STAB   ACIACS   DO IT
00907A F167 86 01    A        LDAA   #1       RBITS=1
00908A F169 97 45    A        STAA   RBITS    SAVE IT
00909A F16B 39                RTS             EXIT
00910                    *
00911                    *      LINE IN ORDER
00912                    *
00913A F16C 7F 0064  A X1     CLR    CTIME2   START TIMER #2
00914A F16F 81 01    A        CMPA   #1       CHECK FOR FIRST BYTE OF MESSAGE LENGTH
00915A F171 26 0E F181        BNE    X2       NO - CHECK AGAIN
00916                    *
00917                    *      XBITS=1 -SEND 1-ST BYTE OF MESSAGE LENGTH
00918                    *
00919A F173 D6 41    A        LDAB   MSGLEN   GET BYTE
00920A F175 F7 6001  A        STAB   ACIADA   SEND IT
00921A F178 D8 43    A        EORB   CHECKS   CALCULATE NEW CHECKSUM
00922A F17A D7 43    A        STAB   CHECKS   SAVE IT
00923A F17C 86 02    A        LDAA   #2       XBITS=2
```

```
PAGE  011  COMM     .SA:1  COMM

00924A F17E 97 46    A         STAA   XBITS     SAVE IT
00925A F180 39                 RTS              EXIT
00926                      *
00927                      *     XBITS NOT 1 - CHECK FOR 2
00928                      *
00929A F181 81 02    A X2      CMPA   #2        CHECK FOR SECOND BYTE OF MESSAGE LENGTH
00930A F183 26 0E F193         BNE    X3        NO - CHECK AGAIN
00931                      *
00932                      *     XBITS=2 -SEND 2-ND BYTE OF MESSAGE LENGTH
00933                      *
00934A F185 D6 42    A         LDAB   MSGLEN+1 GET BYTE
00935A F187 F7 6001  A         STAB   ACIADA    SEND IT
00936A F18A D8 43    A         EORB   CHECKS    CALCULATE NEW CHECKSUM
00937A F18C D7 43    A         STAB   CHECKS    SAVE CHECKSUM
00938A F18E 86 03    A         LDAA   #3        XBITS=3
00939A F190 97 46    A         STAA   XBITS     SAVE IT
00940A F192 39                 RTS              EXIT
00941                      *
00942                      *     XBITS NOT 2 - CHECK FOR 3
00943                      *
00944A F193 81 03    A X3      CMPA   #3        CHECK FOR BODY OF MESSAGE
00945A F195 26 35 F1CC         BNE    X4        NO  - CHECK AGAIN
00946                      *
00947                      *     XBITS=3 -SEND A BYTE FROM DATA BUFFER IF APPLICABLE
00948                      *
00949A F197 DE 49    A         LDX    DLNGTH    CHECK IF ANYTHING IS IN THE BUFFER
00950A F199 27 35 F1D0         BEQ    X41       NOTHING - SEND ERROR CODE
00951A F19B DE 47    A         LDX    DPOINT    GET ADDRESS OF NEXT BYTE
00952A F19D A6 00    A         LDAA   0,X       GET THIS BYTE
00953A F19F B7 6001  A         STAA   ACIADA    SEND IT
00954A F1A2 98 43    A         EORA   CHECKS    UPDATE CHECKSUM
00955A F1A4 97 43    A         STAA   CHECKS    SAVE IT
00956A F1A6 DE 49    A         LDX    DLNGTH    GET LENGTH OF DATA STILL TO BE SENT
00957A F1A8 09                 DEX              DECREMENT IT
00958A F1A9 DF 49    A         STX    DLNGTH    SAVE IT
00959A F1AB 27 07 F1B4         BEQ    X31       CHECK NEXT DATA BUFFER
00960A F1AD DE 47    A         LDX    DPOINT    GET POINTER TO DATA
00961A F1AF 08                 INX              INCREMENT BUFFER POINTER
00962A F1B0 DF 47    A         STX    DPOINT    SAVE IT
00963A F1B2 20 17 F1CB         BRA    X33       GO TO EXIT
00964A F1B4 DE 4D    A X31     LDX    LNGTH2    GET LENGTH OF DATA IN THE SECOND BUFFER
00965A F1B6 26 06 F1BE         BNE    X32
00966A F1B8 86 04    A         LDAA   #4        NO MORE DATA - SET XBITS=4
00967A F1BA 97 46    A         STAA   XBITS     SAVE IT
00968A F1BC 20 0D F1CB         BRA    X33       UPDATE CHECKSUM
00969A F1BE DF 49    A X32     STX    DLNGTH    PUT LENGTH OF SECOND BUFFER INTO THE CURRENT LENGTH BYTE
00970A F1C0 DE 4B    A         LDX    POINT2    GET ADDRESS OF SECOND DATA BUFFER
00971A F1C2 DF 47    A         STX    DPOINT    SAVE IT IN THE CURRENT DATA POINTER
00972A F1C4 CE 0000  A         LDX    #0        ZERO TO X REG
00973A F1C7 DF 4D    A         STX    LNGTH2    CLEAR LENGTH
00974A F1C9 DF 4B    A         STX    POINT2    AND POINTER OF THE SECOND BUFFER
00975A F1CB 39         X33     RTS              EXIT
00976                      *
00977                      *     XBITS NOT 3 - CHECK FOR 4
00978                      *
00979A F1CC 81 04    A X4      CMPA   #4        CHECK FOR ERROR CODE BYTE
00980A F1CE 26 0E F1DE         BNE    X5        NO - CHECK AGAIN
00981                      *
```

```
PAGE  012  COMM    .SA:1  COMM

00982                      *       XBITS=4 -SEND ERROR CODE BYTE
00983                      *
00984A F1D0 D6 44     A X41     LDAB   ERCODE   GET ERROR CODE
00985A F1D2 F7 6001   A         STAB   ACIADA   SEND IT
00986A F1D5 D8 43     A         EORB   CHECKS   UPDATE CHECKSUM
00987A F1D7 D7 43     A         STAB   CHECKS   SAVE IT
00988A F1D9 C6 05     A         LDAB   #5       LOAD NEW VALUE OF XBITS (5)
00989A F1DB D7 46     A         STAB   XBITS    SAVE IT
00990A F1DD 39                  RTS             EXIT
00991                      *
00992                      *       XBITS NOT 4 - CHECK FOR 5
00993                      *
00994A F1DE 81 05     A X5      CMPA   #5       CHECK FOR CHECKSUM BYTE
00995A F1E0 26 0A F1EC          BNE    X6       NO - CHECK AGAIN
00996                      *
00997                      *       XBITS=5 -SEND CHECKSUM BYTE
00998                      *
00999A F1E2 D6 43     A         LDAB   CHECKS   GET CHECKSUM BYTE
01000A F1E4 F7 6001   A         STAB   ACIADA   SEND IT
01001A F1E7 86 06     A         LDAA   #6       SET XBITS=6
01002A F1E9 97 46     A         STAA   XBITS    SAVE IT
01003A F1EB 39                  RTS             EXIT
01004                      *
01005                      *       XBITS NOT 5 - CHECK FOR 6
01006                      *
01007A F1EC 81 06     A X6      CMPA   #6       CHECK FOR 1-ST END CHARACTER
01008A F1EE 26 0A F1FA          BNE    X7       NO - CHECK AGAIN
01009                      *
01010                      *       XBITS=6 -SEND 1-ST END CHARACTER
01011                      *
01012A F1F0 C6 AA     A         LDAB   #%10101010 LOAD CHARACTER INTO ACCUMULATOR
01013A F1F2 F7 6001   A         STAB   ACIADA   SEND IT
01014A F1F5 86 07     A         LDAA   #7       SET XBITS=7
01015A F1F7 97 46     A         STAA   XBITS    SAVE IT
01016A F1F9 39                  RTS             EXIT
01017                      *
01018                      *       XBITS NOT 6 - CHECK FOR 7
01019                      *
01020A F1FA 81 07     A X7      CMPA   #7       CHECK FOR SECOND END CHARACTER
01021A F1FC 26 0A F208          BNE    X8       NO - CHECK AGAIN
01022                      *
01023                      *       XBITS=7 -SEND 2-ND END CHARACTER
01024                      *
01025A F1FE C6 55     A         LDAB   #%01010101 LOAD CHARACTER INTO ACCUMULATOR
01026A F200 F7 6001   A         STAB   ACIADA   SEND IT
01027A F203 86 08     A         LDAA   #8       SET XBITS=8
01028A F205 97 46     A         STAA   XBITS    SAVE IT
01029A F207 39                  RTS             EXIT
01030                      *
01031                      *       XBITS NOT 7 - CHECK FOR 8
01032                      *
01033A F208 81 08     A X8      CMPA   #8       CHECK FOR 3-RD END CHARACTER
01034A F20A 26 0A F216          BNE    X9       NO - CHECK AGAIN
01035                      *
01036                      *       XBITS=8 -SEND 3-RD END CHARACTER
01037                      *
01038A F20C C6 55     A         LDAB   #%01010101 LOAD CHARACTER INTO ACCUMULATOR
01039A F20E F7 6001   A         STAB   ACIADA   SEND IT
```

```
PAGE  013  COMM    .SA:1  COMM

01040A F211 86 09    A        LDAA   #9        SET XBITS=9
01041A F213 97 46    A    .   STAA   XBITS     SAVE IT
01042A F215 39                RTS              EXIT
01043                   *
01044                   *       XBITS NOT 8 - MUST BE 9
01045                   *
01046A F216 C6 55    A X9     LDAB   #%01010101 LOAD CHARACTER INTO ACCUMULATOR
01047A F218 F7 6001  A        STAB   ACIADA    SEND IT
01048A F21B 86 0A    A        LDAA   #10       XBITS=10
01049A F21D 97 46    A        STAA   XBITS     SAVE IT
01050A F21F 39                RTS              EXIT
01051                   *
01052                   *       THE FOLLOWING 14 BYTES ARE RESERVED FOR FURTHER MODIFICATIONS
01053                   *
01054A F220    000E  A        BSZ    14
01055                         END
TOTAL ERRORS 00000--00000
```

```
PAGE  001  EQUATE  .SA:1  PREANS

                           OPT    L
                           NAM    PREANS
                   *.
                   *
                   *        PREPARE ANSWER SUBROUTINE
                   *
                   *
                   *            ***************************************************
                   *            *                                                 *
                   *            *   PROGRAMMER:   W.J. WIERCIENSKI                *
                   *            *                                                 *
                   *            *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS *
                   *            *   TRANSPORTATION TECHNOLOGY AND ENERGY          *
                   *            *   1201 WILSON AVENUE                            *
                   *            *   DOWNSVIEW, ONTARIO  M3M 1J8                   *
                   *            *                                                 *
                   *            *   TELEPHONE (416) 248-7141                      *
                   *            *                                                 *
                   *            *   NOVEMBER  1982 AD                             *
                   *            *                                                 *
                   *            ***************************************************
                   *
                   *
                   *        ANALYSES MESSAGE AND PREPARES EITHER THE ANSWER OR LOGICAL ERROR MESSAGE AS APPLICABLE
                   *
                   *        ON ENTRY TCODE CONTAINS 3-BIT MESSAGE CODE (LEFT JUSTIFIED)
                   *
                   *        THERE ARE EIGHT POSSIBLE MESSAGE CODES FROM BINARY 000 TO 111.
                   *        ONLY SIX OF THEM SHOULD BE SENT FROM CENTRAL TO THE FIELD STATION
                   *        WHILE ALL EIGHT CAN BE RETURN TO CENTRAL.
                   *        THE MESSAGES ARE PROCESSED IN INCREASING ORDER OF CODE AND EACH
                   *        IS DESCRIBED BEFORE IT APPEARS IN THIS ROUTINE.
```

```
PAGE  002  PREANS  .SA:1  PREANS

00532A F240                      ORG    PREANS
00533                    *
00534A F240 D6 44     A          LDAB   ERCODE   GET ERROR CODE
00535                    *
00536                    *        CHECK FOR COMMUNICATION ERRORS
00537                    *
00538A F242 96 4F     A          LDAA   CERROR   CHECK COMMUNICATIONS ERROR COUNT
00539A F244 27 04 F24A           BEQ    P1       NO ERRORS
00540A F246 CA 08     A          ORAB   #%00001000 ERRORS - SET COMMUNICATION ERROR BIT
00541A F248 20 02 F24C           BRA    P2
00542A F24A C4 F7     A P1       ANDB   #%11110111 NO ERRORS - RESET COMMUNICATIONS ERROR BIT
00543                    *
00544                    *        CHECK FOR DETECTOR ERRORS
00545                    *
00546A F24C 96 8C     A P2       LDAA   DETER1   CHECK FOR DETECTOR ERRORS
00547A F24E 26 10 F260           BNE    P3       BRANCH IF FAULTY DETECTORS HAVE BEEN FOUND
00548A F250 96 8D     A          LDAA   DETER2
00549A F252 26 0C F260           BNE    P3       BRANCH IF FAULTY DETECTORS HAVE BEEN FOUND
00550A F254 96 8E     A          LDAA   DETER3
00551A F256 26 08 F260           BNE    P3       BRANCH IF FAULTY DETECTORS HAVE BEEN FOUND
00552A F258 96 8F     A          LDAA   DETER4
00553A F25A 26 04 F260           BNE    P3       BRANCH IF FAULTY DETECTORS HAVE BEEN FOUND
00554A F25C C4 FE     A          ANDB   #%11111110 NO DETECTOR ERRORS, RESET DETECTOR MALFUNCTION BIT
00555A F25E 20 02 F262           BRA    P4       SAVE RESULT
00556A F260 CA 01     A P3       ORAB   #%00000001 SET DETECTOR MALFUNCTION BIT
00557A F262 D7 44     A P4       STAB   ERCODE   SAVE ERROR CODE WORD
00558A F264 7F 0043   A          CLR    CHECKS   CLEAR CHECKSUM
00559A F267 96 40     A          LDAA   TCODE    PUT MESSAGE CODE INTO REGISTER A
```

```
PAGE  003  PREANS  .SA:1  PREANS

00561                    *
00562                    *       CHECK FOR 000 MESSAGE
00563                    *
00564A F269 81 00    A          CMPA    #%00000000 CHECK FOR LOGICAL ERROR
00565A F26B 26 04 F271          BNE     PA1       NO - CHECK AGAIN
00566                    *
00567                    *       MESSAGE CODE = 000 (LOGICAL ERROR - SHOULD NEVER BE SENT)
00568                    *
00569                    *
00570                    *       ON INPUT THIS MESSAGE WILL CAUSE A LOGICAL ERROR MESSAGE TO BE GENERATED.
00571                    *       IT IS USED BY THE FIELD STATION TO ADVISE OF A LOGICAL ERROR IN THE OTHER
00572                    *       MESSAGES. NO BODY OF MESSAGE IS SENT WITH IT.
00573                    *
00574A F26D BD F651  A          JSR     LOGER     PREPARE LOGICAL ERROR MESSAGE
00575A F270 39                  RTS               EXIT
```

```
PAGE  004  PREANS  .SA:1  PREANS

00577                    *
00578                    *      CHECK FOR 001 MESSAGE
00579                    *
00580A F271 81 20    A PA1    CMPA    #%00100000 CHECK FOR 'SEND FIELD STATION MEMORY' REQUEST
00581A F273 26 68 F2DD.       BNE     PA2
00582                    *
00583                    *      MESSAGE CODE = 001  - SEND FIELD STATION MEMORY DATA
00584                    *
00585                    *
00586                    *      USED TO LIST CONTENTS OF MEMORY
00587                    *   ***BODY OF MESSAGE
00588                    *      INPUT
00589                    *        BYTE            CONTENTS
00590                    *          1             STARTING ADDRESS - MSB
00591                    *          2             STARTING ADDRESS - LSB
00592                    *          3             NUMBER OF BYTES TO LIST ( MAX 255 )
00593                    *      OUTPUT
00594                    *        BYTE            CONTENTS
00595                    *          1             ( AS IN
00596                    *          2               INPUT
00597                    *          3               MESSAGE )
00598                    *          4 TO ERROR    CONTENTS OF MEMORY AS REQUESTED
00599                    *            CODE BYTE
00600                    *      THE FOLLOWING CONDITIONS CAN GENERATE A LOGICAL ERROR ANSWER:
00601                    *      - BODY OF MESSAGE INCORRECT
00602                    *      - MESSAGE PARTIALLY OR COMPLETELY OUTSIDE OF RAM
00603                    *      - MESSAGE PARTIALLY OR COMPLETELY OUTSIDE OF PROM
00604                    *
00605A F275 DE 47    A        LDX     DPOINT    GET POINTER TO LAST CHARACTER RECEIVED
00606A F277 8C 0052  A        CPX     #INDATA+2 CHECK FOR CORRECT AMOUNT OF DATA
00607A F27A 26 36 F2B2        BNE     PA12      NOT CORRECT - LOGICAL ERROR
00608A F27C 96 50    A        LDAA    INDATA    GET STARTING ADDRESS OF MEMORY REQUESTED (MSB)
00609A F27E 91 6E    A        CMPA    MRAM      COMPARE WITH END OF RAM (MSB)
00610A F280 22 1D F29F        BHI     PA11      MESSAGE WANTS PROM
00611A F282 26 06 F28A        BNE     PA10      MESSAGE PROBABLY REQUIRES RAM
00612A F284 96 51    A        LDAA    INDATA+1 GET LSB OF STARTING ADDRESS REQUESTED
00613A F286 91 6F    A        CMPA    MRAM+1    CHECK LSB
00614A F288 22 28 F2B2        BHI     PA12      FINAL ADDRESS JUST OUTSIDE OF RAM - LOGICAL ERROR
00615                    *
00616                    *      MESSAGE MAY WANT RAM VALUES - CHECK END ADDRESS
00617                    *
00618A F28A 96 52    A PA10   LDAA    INDATA+2 GET NUMBER OF BYTES REQUESTED
00619A F28C 4A                DECA              SUBTRACT 1 (TO FIND ADDRESS OF FINAL BYTE
00620A F28D 9B 51    A        ADDA    INDATA+1 ADD LSB OF STARTING ADDRESS REQUESTED
00621A F28F D6 50    A        LDAB    INDATA    GET MSB OF STARTING ADDRESS REQUESTED
00622A F291 C9 00    A        ADCB    #0        ADD IN CARRY IF ANY
00623A F293 D1 6E    A        CMPB    MRAM      COMPARE WITH MSB OF MEMORY AVAILABLE
00624A F295 22 1B F2B2        BHI     PA12      GREATER THEN END OF RAM - LOGICAL ERROR
00625A F297 26 1D F2B6        BNE     PA13      SIZE OF MEMORY REQUESTED IS OKAY
00626A F299 91 6F    A        CMPA    MRAM+1    COMPARE WITH LSB OF MEMORY AVAILABLE
00627A F29B 22 15 F2B2        BHI     PA12      MEMORY REQUESTED IS OUT OF RANGE AVAILABLE
00628A F29D 20 17 F2B6        BRA     PA13      OKAY - GO TO PREPARE THE RETURN MESSAGE
00629                    *
00630                    *      MESSAGE MAY WANT PROM
00631                    *
00632A F29F 96 50    A PA11   LDAA    INDATA    GET MSB OF STARTING ADDRESS
00633A F2A1 81 C0    A        CMPA    #$C0      COMPARE STARTING ADDRESS WITH BEGINNING OF PROM
00634A F2A3 22 02 F2A7        BHI     PA11X     CHECK FOR END ADDRESS
```

```
PAGE  005  PREANS  .SA:1  PREANS

00635A F2A5 26 0B F2B2        BNE   PA12     LOGICAL ERROR
00636A F2A7 96 52     A PA11X LDAA  INDATA+2 GET NUMBER OF BYTES REQUESTED
00637A F2A9 4A                DECA           DECREMENT TO FIND ADDRESS OF FINAL BYTE
00638A F2AA 9B 51     A       ADDA  INDATA+1 ADD LSB OF STARTING ADDRESS REQUESTED
00639A F2AC D6 50     A       LDAB  INDATA   GET MSB OF STARTING ADDRESS REQUESTED
00640A F2AE C9 00     A       ADCB  #0       ADD IN THE CARRY IF ANY
00641A F2B0 24 04 F2B6        BCC   PA13     OK EXISTS IN PROM (MAX $FFFF)
00642A F2B2 BD F651   A PA12  JSR   LOGER    NOT OKAY - PREPARE LOGICAL ERROR MESSAGE
00643A F2B5 39                RTS            RETURN TO COMMUNICATIONS SUBROUTINE
00644                   *
00645                   *       LIMITS OKAY - PREPARE MESSAGE
00646                   *
00647A F2B6 96 00     A PA13  LDAA  PCSCAD   GET ADDRESS
00648A F2B8 8B 20     A       ADDA  #%00100000 ADD MESSAGE CODE
00649A F2BA 97 40     A       STAA  TCODE    SAVE IT
00650A F2BC 5F                CLRB           REG B=0
00651A F2BD 96 52     A       LDAA  INDATA+2 GET LENGTH OF DATA REQUESTED
00652A F2BF 8B 08     A       ADDA  #8       ADD REST OF MESSAGE
00653A F2C1 C9 00     A       ADCB  #0       ADD CARRY BIT
00654A F2C3 D7 41     A       STAB  MSGLEN   (SAVE LENGTH
00655A F2C5 97 42     A       STAA  MSGLEN+1 OF MESSAGE)
00656A F2C7 CE 0050   A       LDX   #INDATA  BEGINNING OF BUFFER #1
00657A F2CA DF 47     A       STX   DPOINT   SAVE IT
00658A F2CC CE 0003   A       LDX   #3       LENGTH OF BUFFER #1
00659A F2CF DF 49     A       STX   DLNGTH   SAVE IT
00660A F2D1 DE 50     A       LDX   INDATA   BEGINNING OF MEMORY REQUESTED
00661A F2D3 DF 4B     A       STX   POINT2   SAVE IT
00662A F2D5 96 52     A       LDAA  INDATA+2 GET LENGTH
00663A F2D7 7F 004D   A       CLR   LNGTH2   MSB OF LENGTH
00664A F2DA 97 4E     A       STAA  LNGTH2+1 LSB OF LENGTH
00665A F2DC 39                RTS
```

```
PAGE  006  PREANS  .SA:1  PREANS

00667                      *
00668                      *        CHECK FOR 010 MESSAGE
00669                      *
00670A F2DD 81 40     A PA2    CMPA   #%01000000 CHECK FOR NEGATIVE ACKNOWLEDGEMENT MESSAGE
00671A F2DF 26 04 F2E5         BNE    PA3       NO - CHECK AGAIN
00672                      *
00673                      *       MESSAGE CODE=010  - NEGATIVE ACKNOWLEDGEMENT MESSAGE  (SHOULD NEVER BE SENT)
00674                      *
00675                      *
00676                      *       ON INPUT THIS MESSAGE WILL CAUSE A LOGICAL ERROR MESSAGE TO BE GENERATED.
00677                      *       IT IS USED BY THE RECEIVING SUBROUTINE TO INDICATE INCORRECT FORMAT OF MESSAGE,
00678                      *       INPUT ERROR, OR CHECKSUM ERROR. NO BODY OF MESSAGE IS SENT WITH IT.
00679                      *
00680A F2E1 BD F651  A        JSR    LOGER     PREPARE LOGICAL ERROR MESSAGE
00681A F2E4 39                RTS              EXIT
```

```
PAGE  007  PREANS  .SA:1  PREANS

00683                    *
00684                    *        CHECK FOR 011 MESSAGE
00685                    *
00686A F2E5 81 60    A PA3    CMPA   #%01100000 CHECK FOR HOURLY VOLUME COUNT MESSAGE
00687A F2E7 26 61 F34A        BNE    PA4        NO - TRY AGAIN
00688                    *
00689                    *        MESSAGE CODE=011  - SEND HOURLY VOLUME COUNT DATA
00690                    *
00691                    *
00692                    *        USED TO OBTAIN HOURLY VOLUME COUNTS FOR A SPECIFIC DAY
00693                    *     ***BODY OF MESSAGE
00694                    *        INPUT
00695                    *          BYTE             CONTENTS
00696                    *            1               YEAR OF CENTURY ( BCD )
00697                    *            2               MONTH OF YEAR ( BCD )
00698                    *            3               DAY OF MONTH ( BCD )
00699                    *            4               RESET CONTROL BYTE ( DESCRIBED IN RESCON SUBROUTINE )
00700                    *        OUTPUT
00701                    *          BYTE               CONTENTS
00702                    *            1                ( AS IN
00703                    *            2                  INPUT
00704                    *            3                  MESSAGE )
00705                    *            4 TO ERROR       THE HOURLY VOLUME COUNT DATA FOR THE DAY REQUESTED
00706                    *              CODE BYTE
00707                    *        THE FOLLOWING CONDITIONS CAN CAUSE A LOGICAL ERROR ANSWER:
00708                    *        - BODY OF INPUT MESSAGE WAS INCORRECT
00709                    *        - DATA IS NOT AVAILABLE FOR THE DAY REQUESTED
00710                    *          ( MAY ALSO BE CAUSED BY A POINTER ERROR )
00711                    *
00712A F2E9 DE 47      A      LDX    DPOINT     GET POINTER TO THE LAST CHARACTER RECEIVED
00713A F2EB 8C 0053   A      CPX    #INDATA+3 CHECK FOR CORRECT AMOUNT OF DATA
00714A F2EE 26 33 F323        BNE    PA33       NOT CORRECT - LOGICAL ERROR
00715A F2F0 96 70      A      LDAA   VOLDAY     GET # OF DAYS VOLUME DATA AVAILABLE
00716A F2F2 27 2F F323        BEQ    PA33       NO VOLUME DATA AVAILABLE - LOGICAL ERROR
00717A F2F4 91 7F      A      CMPA   DAYMAX     CHECK FOR MAXIMUM PERMISSABLE
00718A F2F6 22 2B F323        BHI    PA33       LOGICAL ERROR
00719                    *
00720                    *        TRY TO FIND DATA REQUESTED BY COMPARING DATES AVAILABLE
00721                    *
00722A F2F8 DE 71      A      LDX    VODAY1     ADDRESS OF FIRST DAY VOLUME DATA
00723A F2FA D6 50      A PA31 LDAB   INDATA+0 GET YEAR REQUESTED
00724A F2FC E1 00      A      CMPB   0,X        COMPARE WITH YEAR AVAILABLE
00725A F2FE 26 0E F30E        BNE    PA32       NOT THE SAME - TRY AGAIN
00726A F300 D6 51      A      LDAB   INDATA+1 GET MONTH REQUESTED
00727A F302 E1 01      A      CMPB   1,X        COMPARE WITH MONTH AVAILABLE
00728A F304 26 08 F30E        BNE    PA32       NOT THE SAME - TRY AGAIN
00729A F306 D6 52      A      LDAB   INDATA+2 GET DAY REQUESTED
00730A F308 E1 02      A      CMPB   2,X        COMPARE WITH DAY AVAILABLE
00731A F30A 26 02 F30E        BNE    PA32       NOT THE SAME - TRY AGAIN
00732A F30C 20 19 F327        BRA    PA34       DATE AGREES - PREPARE MESSAGE
00733A F30E 4A           PA32 DECA              CALCULATE # OF DAYS STILL TO SEARCH
00734A F30F 27 12 F323        BEQ    PA33       NOTHING LEFT - LOGICAL ERROR
00735A F311 DF 3E      A      STX    TEMP01     STORE POINTER TO VOLUME DATA IN TEMPORY LOCATION
00736A F313 D6 3F      A      LDAB   TEMP02     GET LSB OF ADDRESS
00737A F315 DB 74      A      ADDB   VOBUF1+1 ADD LSB OF LENGTH
00738A F317 D7 3F      A      STAB   TEMP02     SAVE IT
00739A F319 D6 3E      A      LDAB   TEMP01     GET MSB OF ADDRESS
00740A F31B D9 73      A      ADCB   VOBUF1+0 ADD MSB OF LENGTH AND CARRY IF ANY
```

```
PAGE  006  PREANS  .SA:1  PREANS

00741A F31D D7 3E     A          STAB   TEMP01   SAVE IT
00742A F31F DE 3E     A          LDX    TEMP01   NEW ADDRESS
00743A F321 20 D7 F2FA           BRA    PA31     TRY AGAIN
00744A F323 BD F651   A PA33     JSR    LOGER    PREPARE LOGICAL ERROR MESSAGE
00745A F326 39                   RTS             EXIT
00746                  *
00747                  *          OKAY - PREPARE VOLUME COUNT MESSAGE  - REGISTER X CONTAINS POINTER TO DATA
00748                  *
00749A F327 D6 00     A PA34     LDAB   PCSCAD   GET STATION ADDRESS
00750A F329 CB 60     A          ADDB   #%01100000 ADD MESSAGE CODE
00751A F32B D7 40     A          STAB   TCODE    SAVE IT
00752                  *
00753                  *          CALCULATE LENGTH OF MESSAGE
00754                  *
00755A F32D D6 74     A          LDAB   VOBUF1+1 GET LSB OF LENGTH
00756A F32F CB 05     A          ADDB   #5       ADD 5
00757A F331 D7 42     A          STAB   MSGLEN+1 SAVE IT
00758A F333 D6 73     A          LDAB   VOBUF1+0 GET MSB OF LENGTH
00759A F335 C9 00     A          ADCB   #0       ADD CARRY IF ANY
00760A F337 D7 41     A          STAB   MSGLEN+0 SAVE IT
00761                  *
00762                  *          PROCESS RESET CONTROL AND PREPARE POINTERS
00763                  *
00764A F339 BD F5EB   A          JSR    RESCON   PROCESS RESET CONTROL
00765A F33C DF 47     A          STX    DPOINT   POINTER TO DATA
00766A F33E DE 73     A          LDX    VOBUF1   GET LENGTH OF DATA
00767A F340 DF 49     A          STX    DLNGTH   STORE IT
00768A F342 CE 0000   A          LDX    #0       CLEAR X-REGISTER
00769A F345 DF 4B     A          STX    POINT2   CLEAR SECOND BUFFER
00770A F347 DF 4D     A          STX    LNGTH2   AND LENGTH
00771A F349 39                   RTS             RETURN
```

```
PAGE  009  PREANS  .SA:1  PREANS

00773                      *
00774                      *
00775                      *      CHECK FOR 100 MESSAGE
00776A F34A 81 80      A PA4     CMPA   #%10000000 CHECK FOR LENGTH COUNT AND VEHICLE SPEED DATA MESSAGE
00777A F34C 27 03 F351           BEQ    PA40      GO TO PROCESS MESSAGE
00778A F34E 7E F3F1    A         JMP    PA5       OKAY
00779                      *
00780                      *
00781                      *      MESSAGE CODE=100  - SEND HOURLY LENGTH COUNT AND VEHICLE SPEED DATA
00782                      *
00783                      *
00784                      *      USED TO OBTAIN HOURLY LENGTH COUNT DATA AND VEHICLE SPEED DATA
00785                      *      FOR A GIVEN DAY.
00786                      *   ***BODY OF MESSAGE
00787                      *      INPUT
00788                      *        BYTE              CONTENTS
00789                      *          1               YEAR OF CENTURY ( BCD )
00790                      *          2               MONTH OF YEAR ( BCD )
00791                      *          3               DAY OF MONTH ( BCD )
00792                      *          4               RESET CONTROL BYTE ( DESCRIBED IN RESCON SUBROUTINE )
00793                      *      OUTPUT
00794                      *       BYTE               CONTENTS
00795                      *          1               ( AS IN
00796                      *          2                 INPUT
00797                      *          3                 MESSAGE )
00798                      *          4 TO ERROR      THE HOURLY LENGTH COUNT AND SPEED DATA FOR THE DAY REQUESTED
00799                      *            CODE BYTE
00800                      *      THE FOLLOWING CONDITIONS CAN GENERATE A LOGICAL ERROR ANSWER:
00801                      *      - BODY OF MESSAGE INCORRECT
00802                      *      - LENGTH DATA AND/OR SPEED DATA NOT AVAILABLE FOR THE DAY
00803                      *        REQUESTED. ( MAY ALSO BE CAUSED BY A POINTER ERROR )
00804A F351 DE 47      A PA40    LDX    DPOINT    GET POINTER TO LAST CHARACTER RECEIVED
00805A F353 8C 0053    A         CPX    #INDATA+3 CHECK FOR CORRECT AMOUNT OF DATA
00806A F356 26 35 F38D           BNE    PA43      NOT CORRECT - LOGICAL ERROR
00807A F358 96 7A      A         LDAA   LENDAY    GET # OF DAYS LENGTH DATA AVAILABLE
00808A F35A 27 31 F38D           BEQ    PA43      NO LENGTH DATA AVAILABLE - LOGICAL ERROR
00809A F35C 91 7F      A         CMPA   DAYMAX    CHECK FOR MAXIMUM PERMISSABLE
00810A F35E 22 2D F38D           BHI    PA43      LOGICAL ERROR
00811A F360 DE 7B      A         LDX    LNDAY1    GET POINTER TO THE FIRST DAY'S LENGTH DATA
00812                      *
00813                      *
00814                      *      TRY TO FIND LENGTH DATA REQUESTED BY COMPARING DATES AVAILABLE
00815A F362 D6 50      A PA41    LDAB   INDATA+0 GET YEAR REQUESTED
00816A F364 E1 00      A         CMPB   0,X       COMPARE WITH YEAR AVAILABLE
00817A F366 26 10 F378           BNE    PA42      NOT THE SAME - TRY AGAIN
00818A F368 D6 51      A         LDAB   INDATA+1 GET MONTH REQUESTED
00819A F36A E1 01      A         CMPB   1,X       COMPARE WITH MONTH AVAILABLE
00820A F36C 26 0A F378           BNE    PA42      NOT THE SAME - TRY AGAIN
00821A F36E D6 52      A         LDAB   INDATA+2 GET DAY REQUESTED
00822A F370 E1 02      A         CMPB   2,X       COMPARE WITH DAY AVAILABLE
00823A F372 26 04 F378           BNE    PA42      NOT THE SAME - TRY AGAIN
00824A F374 DF 47      A         STX    DPOINT    LENGTH DATA FOUND - SAVE POINTER
00825A F376 20 19 F391           BRA    PA44      CHECK FOR AVAILABILITY OF SPEED DATA
00826A F378 4A           PA42    DECA             DECREMENT NUMBER OF DAYS LENGTH DATA AVAILABLE
00827A F379 27 12 F38D           BEQ    PA43      DATE NOT FOUND - LOGICAL ERROR
00828A F37B DF 3E      A         STX    TEMP01    STORE POINTER TO LENGTH DATA IN A TEMPORY LOCATION
00829A F37D D6 3F      A         LDAB   TEMP02    GET LSB OF ADDRESS
00830A F37F DB 7E      A         ADDB   LNBUF1+1 ADD IN LSB OF LENGTH OF THE LENGTH DATA
```

```
PAGE  010  PREANS  .SA:1  PREANS

00831A F381 D7 3F     A          STAB   TEMP02    SAVE IT
00832A F383 D6 3E     A          LDAB   TEMP01    GET MSB OF ADDRESS
00833A F385 D9 7D     A          ADCB   LNBUF1+0  ADD IN MSB OF LENGTH AND CARRY IF ANY
00834A F387 D7 3E     A          STAB   TEMP01    SAVE IT
00835A F389 DE 3E     A          LDX    TEMP01    GET ADDRESS TO THE NEXT DAY'S LENGTH DATA
00836A F38B 20 D5 F362           BRA    PA41      TRY AGAIN
00837A F38D BD F651   A PA43     JSR    LOGER     GO TO PREPARE LOGICAL ERROR MESSAGE
00838A F390 39                   RTS              EXIT TO COMMUNICATIONS
00839                   *
00840                   *          CHECK FOR AVAILABILITY OF SPEED DATA
00841                   *
00842A F391 96 75     A PA44     LDAA   SPDDAY    GET NUMBER OF DAYS DATA AVAILABLE
00843A F393 27 31 F3C6           BEQ    PA47      NO SPEED DATA AVAILABLE - LOGICAL ERROR
00844A F395 91 7F     A          CMPA   DAYMAX    CHECK FOR MAXIMUM PERMISSABLE
00845A F397 22 2D F3C6           BHI    PA47      LOGICAL ERROR
00846A F399 DE 76     A          LDX    SPDAY1    GET POINTER TO THE FIRST DAY'S SPEED DATA
00847                   *
00848                   *          TRY TO FIND SPEED DATA REQUESTED BY COMPARING DATES AVAILABLE
00849                   *
00850A F39B D6 50     A PA45     LDAB   INDATA+0  GET YEAR REQUESTED
00851A F39D E1 00     A          CMPB   0,X       COMPARE WITH YEAR AVAILABLE
00852A F39F 26 10 F3B1           BNE    PA46      NOT THE SAME - TRY AGAIN
00853A F3A1 D6 51     A          LDAB   INDATA+1  GET MONTH REQUESTED
00854A F3A3 E1 01     A          CMPB   1,X       COMPARE WITH MONTH AVAILABLE
00855A F3A5 26 0A F3B1           BNE    PA46      NOT THE SAME - TRY AGAIN
00856A F3A7 D6 52     A          LDAB   INDATA+2  GET DAY REQUESTED
00857A F3A9 E1 02     A          CMPB   2,X       COMPARE WITH DAY AVAILABLE
00858A F3AB 26 04 F3B1           BNE    PA46      NOT THE  SAME - TRY AGAIN
00859A F3AD DF 4B     A          STX    POINT2    SPEED DATA FOUND - SAVE POINTER
00860A F3AF 20 1E F3CF           BRA    PA48      DATA FOUND - PROCESS THE REST OF THE MESSAGE
00861                   *
00862                   *          CALCULATE POINTER TO THE NEXT DAY'S SPEED DATA
00863                   *
00864A F3B1 4A          PA46     DECA             DECREMENT NUMBER OF DAYS SPEED DATA AVAILABLE
00865A F3B2 27 12 F3C6           BEQ    PA47      DATE NOT FOUND - LOGICAL ERROR
00866A F3B4 DF 3E     A          STX    TEMP01    STORE POINTER TO SPEED DATA IN A TEMPORY LOCATION
00867A F3B6 D6 3F     A          LDAB   TEMP02    GET LSB OF ADDRESS
00868A F3B8 DB 79     A          ADDB   SPBUF+1   ADD LSB OF LENGTH OF THE SPEED DATA
00869A F3BA D7 3F     A          STAB   TEMP02    SAVE IT
00870A F3BC D6 3E     A          LDAB   TEMP01    GET MSB OF ADDRESS
00871A F3BE D9 78     A          ADCB   SPBUF+0   ADD MSB OF LENGTH AND THE CARRY IF ANY
00872A F3C0 D7 3E     A          STAB   TEMP01    SAVE IT
00873A F3C2 DE 3E     A          LDX    TEMP01    GET ADDRESS TO THE NEXT DAY'S SPEED DATA
00874A F3C4 20 D5 F39B           BRA    PA45      TRY AGAIN
00875A F3C6 CE 0000   A PA47     LDX    #0        CLEAR X-REGISTER
00876A F3C9 DF 47     A          STX    DPOINT    CLEAR POINTER TO THE FIRST DATA BUFFER
00877A F3CB BD F651   A          JSR    LOGER     PREPARE LOGICAL ERROR MESSAGE
00878A F3CE 39                   RTS              EXIT
00879                   *
00880                   *          PROCESS REMAINDER OF MESSAGE
00881                   *
00882A F3CF DE 7D     A PA48     LDX    LNBUF1    GET LENGTH OF THE LENGTH DATA
00883A F3D1 DF 49     A          STX    DLNGTH    SAVE IT FOR TRANSMISSION
00884A F3D3 DE 78     A          LDX    SPBUF     GET LENGTH OF SPEED DATA
00885A F3D5 DF 4D     A          STX    LNGTH2    SAVE IT FOR TRANSMISSION
00886A F3D7 D6 00     A          LDAB   PCSCAD    GET STATION ADDRESS
00887A F3D9 CB 80     A          ADDB   #%10000000 ADD IN MESSAGE CODE
00888A F3DB D7 40     A          STAB   TCODE
```

```
PAGE  011  PREANS  .SA:1  PREANS

00889                    *
00890                    *  .    CALCULATE LENGTH OF MESSAGE
00891                    *
00892A F3DD D6 7E    A          LDAB   LNBUF1+1 GET LSB OF LENGTH OF THE LENGTH DATA
00893A F3DF DB 79    A          ADDB   SPBUF+1  ADD LSB OF LENGTH OF SPEED DATA
00894A F3E1 96 7D    A          LDAA   LNBUF1+0 GET MSB OF LENGTH OF THE LENGTH DATA
00895A F3E3 99 78    A          ADCA   SPBUF+0  ADD MSB OF SPEED DATA AND THE CARRY IF ANY
00896A F3E5 CB 05    A          ADDB   #5       ADD IN 5 FOR INFO DATA BYTES
00897A F3E7 89 00    A          ADCA   #0       ADD IN THE CARRY IF ANY
00898A F3E9 97 41    A          STAA   MSGLEN   SAVE MSB OF MESSAGE LENGTH FOR TRANSMISSION
00899A F3EB D7 42    A          STAB   MSGLEN+1 SAVE LSB OF MESSAGE LENGTH FOR TRANSMISSION
00900A F3ED BD F5EB  A          JSR    RESCON   GO TO PROCESS RESET CONTROL
00901A F3F0 39                  RTS             EXIT TO COMMUNICATIONS
```

```
00903                           *
00904                           *      CHECK FOR 101 MESSAGE
00905                           *
00906A F3F1 81 A0      A PA5     CMPA   #%10100000 CHECK FOR START SPEED AND LENGTH DATA ACCUMULATION MESSAGE
00907A F3F3 26 69 F45E .         BNE    PA6       NO - CHECK AGAIN
00908                           *
00909                           *      MESSAGE CODE=101  - START VEHICLE SPEED AND LENGTH DATA ACCUMULATION
00910                           *
00911                           *
00912                           *      USED TO START VEHICLE SPEED AND LENGTH DATA ACCUMULATION AT 00:00 HOUR
00913                           *      ON THE DAY REQUESTED.
00914                           *   ***BODY OF MESSAGE
00915                           *      INPUT
00916                           *        BYTE            CONTENTS
00917                           *          1             YEAR OF CENTURY ( BCD )
00918                           *          2             MONTH OF YEAR ( BCD )
00919                           *          3             DAY OF MONTH ( BCD )
00920                           *          4             RESET CONTROL BYTE ( DESCRIBED IN RESCON SUBROUTINE )
00921                           *      OUTPUT
00922                           *        BYTE            CONTENTS
00923                           *          1             ( AS IN
00924                           *          2               INPUT
00925                           *          3               MESSAGE )
00926                           *      THE FOLLOWING CONDITIONS WILL GENERATE A LOGICAL ERROR ANSWER:
00927                           *      - INCORRECT BODY OF MESSAGE
00928                           *      - DATE REQUESTED LESS THAN OR EQUAL CURRENT DATE
00929                           *      - DATE REQUESTED IS FEBRUARY 29TH AND IT IS NOT A LEAP YEAR
00930                           *
00931A F3F5 DE 47      A         LDX    DPOINT    GET POINTER TO THE LAST CHARACTER RECEIVED
00932A F3F7 8C 0053    A         CPX    #INDATA+3 CHECK FOR CORRECT AMOUNT OF DATA
00933A F3FA 26 30 F42C           BNE    PA52      NOT CORRECT - LOGICAL ERROR
00934                           *
00935                           *       CHECK YEAR
00936                           *
00937A F3FC 96 50      A         LDAA   INDATA+0 GET YEAR REQUESTED IN THE MESSAGE
00938A F3FE 91 03      A         CMPA   YEAR      COMPARE WITH VALUE OF CURRENT YEAR
00939A F400 22 10 F412           BHI    PA50      YES GIVEN YEAR IS GREATER THAN CURRENT YEAR
00940A F402 26 28 F42C           BNE    PA52      LOGICAL ERROR
00941                           *
00942                           *       CHECK MONTH
00943                           *
00944A F404 96 51      A         LDAA   INDATA+1 GET MONTH REQUESTED IN THE MESSAGE
00945A F406 91 04      A         CMPA   MONTH     COMPARE WITH VALUE OF CURRENT MONTH
00946A F408 22 08 F412           BHI    PA50      GIVEN DATE IS GREATER THAN CURRENT DATE
00947A F40A 26 20 F42C           BNE    PA52      LOGICAL ERROR
00948                           *
00949                           *       CHECK DAY
00950                           *
00951A F40C D6 52      A         LDAB   INDATA+2 GET DAY REQUESTED IN THE MESSAGE
00952A F40E D1 05      A         CMPB   DATE+2    COMPARE WITH CURRENT DAY
00953A F410 23 1A F42C           BLS    PA52      IMPOSSIBLE - LOGICAL ERROR
00954                           *
00955                           *      CHECK FOR LEAP YEAR CASE
00956                           *
00957A F412 D6 51      A PA50    LDAB   INDATA+1 GET GIVEN MONTH
00958A F414 C1 02      A         CMPB   #2        SEE IF IT IS FEBRUARY
00959A F416 26 18 F430           BNE    PA53      NO IT IS NOT
00960A F418 D6 52      A         LDAB   INDATA+2 GET GIVEN DAY
```

```
PAGE  013  PREANS  .SA:1  PREANS

00961A F41A C1 29     A          CMPB   #$29     SEE IF IT IS THE 29TH
00962A F41C 26 12 F430           BNE    PA53     NO IT IS NOT
00963                      *
00964                      *       CHECK FOR LEAP YEAR
00965                      *
00966A F41E 4F                   CLRA            SET A=0
00967A F41F 5F                   CLRB            SET B=0
00968A F420 8B 01     A PA51     ADDA   #1       A=A+1
00969A F422 19                   DAA             CONVERT TO BCD FORMAT
00970A F423 5C                   INCB            B=B+1
00971A F424 91 50     A          CMPA   INDATA   COMPARE YEAR WITH GIVEN YEAR
00972A F426 26 F8 F420           BNE    PA51     NO IT IS NOT
00973A F428 C5 03     A          BITB   #%00000011 IS IT A LEAP YEAR?
00974A F42A 27 04 F430           BEQ    PA53     YES - IT IS LEAP YEAR
00975                      *
00976                      *       LOGICAL ERROR
00977                      *
00978A F42C BD F651   A PA52     JSR    LOGER    PREPARE LOGICAL ERROR MESSAGE
00979A F42F 39                   RTS             RETURN TO COMMUNICATIONS
00980                      *
00981                      *       PREPARE RESPONSE MESSAGE FOR START VEHICLE SPEED AND LENGTH DATA ACCUMULATION
00982                      *
00983A F430 D6 6C     A PA53     LDAB   MFLAG    GET BYTE CONTAINING START SPEED AND LENGTH DATA ACCUMULATION FLAG
00984A F432 CA 01     A          ORAB   #%00000001 SET THE FLAG
00985A F434 D7 6C     A          STAB   MFLAG    SAVE IT
00986A F436 DE 50     A          LDX    INDATA   GET GIVEN YEAR AND MONTH
00987A F438 DF 66     A          STX    SDATE    SAVE IT
00988A F43A D6 52     A          LDAB   INDATA+2 GET GIVEN DAY
00989A F43C D7 68     A          STAB   SDAY     SAVE IT
00990A F43E BD F5EB   A          JSR    RESCON   GO TO PROCESS RESET CONTROL
00991A F441 D6 00     A          LDAB   PCSCAD   GET STATION ADDRESS
00992A F443 CB A0     A          ADDB   #%10100000 ADD IN THE TRANSMISSION CODE
00993A F445 D7 40     A          STAB   TCODE    SAVE IT FOR TRANSMISSION
00994A F447 CE 0008   A          LDX    #8       ACCUMULATOR B CONTAINS 8
00995A F44A DF 41     A          STX    MSGLEN   SAVE THE MESSAGE LENGTH
00996A F44C CE 0050   A          LDX    #INDATA  GET POINTER TO THE BEGINNING OF THE OUTPUT BUFFER
00997A F44F DF 47     A          STX    DPOINT   SAVE IT
00998A F451 CE 0003   A          LDX    #3       SET X-REGISTER EQUAL TO THREE
00999A F454 DF 49     A          STX    DLNGTH   SAVE IT
01000A F456 CE 0000   A          LDX    #0       SET X-REGISTER EQUAL TO ZERO
01001A F459 DF 4B     A          STX    POINT2   CLEAR SECOND BUFFER POINTER
01002A F45B DF 4D     A          STX    LNGTH2   CLEAR SECOND BUFFER LENGTH
01003A F45D 39                   RTS             RETURN TO COMMUNICATIONS
```

```
01005                      *
01006                      *       CHECK FOR 110 MESSAGE
01007                      *
01008A F45E 81 C0    A PA6    CMPA   #%11000000 CHECK FOR END SPEED AND LENGTH DATA ACCUMULATION MESSAGE
01009A F460 27 03 F465        BEQ    JP         OKAY
01010A F462 7E F4F4  A        JMP    PA7        NO - CHECK AGAIN
01011                      *
01012                      *       MESSAGE CODE=110  - END VEHICLE SPEED AND LENGTH DATA ACCUMULATION
01013                      *
01014                      *
01015                      *       USED TO STOP VEHICLE SPEED AND LENGTH DATA ACCUMULATION AT 24:00 HOURS
01016                      *       ON THE DAY REQUESTED.
01017                      *    ***BODY OF MESSAGE
01018                      *       INPUT
01019                      *         BYTE             CONTENTS
01020                      *           1              YEAR OF CENTURY ( BCD )
01021                      *           2              MONTH OF CENTURY ( BCD )
01022                      *           3              DAY OF MONTH ( BCD )
01023                      *           4              RESET CONTROL BYTE ( DESCRIBED IN RESCON SUBROUTINE )
01024                      *       OUTPUT
01025                      *         BYTE             CONTENTS
01026                      *           1              ( AS IN
01027                      *           2                INPUT
01028                      *           3                MESSAGE )
01029                      *       THE FOLLOWING CONDITIONS WILL GENERATE A LOGICAL ERROR MESSAGE:
01030                      *       - INCORRECT BODY OF MESSAGE
01031                      *       - SPEED/LENGTH CALCULATION IN PROGRESS BUT REQUEST DATE ALREADY PASSED
01032                      *       - SPEED/LENGTH CALCULATION NOT IN PROGRESS & NO REQUEST TO START FOUND
01033                      *       - SPEED/LENGTH CALCULATION NOT IN PROGRESS & END DATE GREATER THAN OR EQUAL
01034                      *         REQUESTED STARTING DATE
01035                      *       [illegible]TE REQUESTED IS FEBRUARY 29TH AND IT IS NOT A LEAP YEAR
01036                      *
01037A F465 DE 47    A JP     LDX    DPOINT     GET POINTER TO THE LAST CHARACTER RECEIVED
01038A F467 8C 0053  A        CPX    #INDATA+3 CHECK FOR CORRECT AMOUNT OF DATA
01039A F46A 26 56 F4C2        BNE    PA63       NOT CORRECT - LOGICAL ERROR
01040A F46C D6 75    A        LDAB   SPDDAY     GET NUMBER OF DAYS SPEED DATA COLLECTED
01041A F46E 26 20 F490        BNE    PA60       YES - THERE IS SPEED DATA
01042A F470 D6 6C    A        LDAB   MFLAG      GET BYTE CONTAINING SPEED CALCULATION BIT
01043A F472 C5 01    A        BITB   #%00000001 CHECK START SPEED CALCULATION FLAG
01044A F474 27 4C F4C2        BEQ    PA63       NO - LOGICAL ERROR
01045                      *
01046                      *        CHECK YEAR
01047                      *
01048A F476 96 50    A        LDAA   INDATA     YEAR GIVEN
01049A F478 91 66    A        CMPA   SYEAR      COMPARE WITH STARTING YEAR
01050A F47A 22 2C F4A8        BHI    PA61       OKAY
01051A F47C 26 44 F4C2        BNE    PA63       LOGICAL ERROR
01052                      *
01053                      *       CHECK MONTH
01054                      *
01055A F47E 96 51    A        LDAA   INDATA+1 GET MONTH GIVEN
01056A F480 91 67    A        CMPA   SMONTH     COMPARE WITH STARTING MONTH
01057A F482 22 24 F4A8        BHI    PA61       OKAY
01058A F484 26 3C F4C2        BNE    PA63       LOGICAL ERROR
01059                      *
01060                      *        CHECK DAY
01061                      *
01062A F486 D6 52    A        LDAB   INDATA+2 GET DAY FOR ENDING SPEED CALCULATIONS
```

```
01063A F488 D1 68     A        CMPB   SDAY      COMPARE WITH DAY FOR STARTING SPEED CALCULATION
01064A F48A 22 1C F4A8         BHI    PA61      OKAY SO FAR
01065A F48C 27 1A F4A8         BEQ    PA61
01066A F48E 20 32 F4C2         BRA    PA63      NO - LOGICAL ERROR
01067                      *
01068                      *        SPEED DATA HAS BEEN COLLECTED. CHECK TO SEE IF IT IS TO BE
01069                      *        TERMINATED TODAY OR LATER. I.E. NO LOGICAL ERRORS.
01070                      *
01071                      *
01072                      *         CHECK YEAR
01073                      *
01074A F490 96 50     A PA60   LDAA   INDATA    YEAR GIVEN
01075A F492 91 03     A        CMPA   YEAR      COMPARE WITH CURRENT YEAR
01076A F494 22 12 F4A8         BHI    PA61      OKAY
01077A F496 26 2A F4C2         BNE    PA63      LOGICAL ERROR
01078                      *
01079                      *         CHECK  MONTH
01080                      *
01081A F498 96 51     A        LDAA   INDATA+1 GET MONTH GIVEN
01082A F49A 91 04     A        CMPA   MONTH     COMPARE WITH CURRENT MONTH
01083A F49C 22 0A F4A8         BHI    PA61      OKAY
01084A F49E 26 22 F4C2         BNE .  PA63      LOGICAL ERROR
01085                      *
01086                      *         CHECK DAY
01087                      *
01088A F4A0 D6 52     A        LDAB   INDATA+2 GET DAY FOR ENDING CALCULATIONS
01089A F4A2 D1 05     A        CMPB   DAOFMO    COMPARE CURRENT DAY WITH GIVEN DAY
01090A F4A4 22 02 F4A8         BHI    PA61      OKAY
01091A F4A6 26 1A F4C2         BNE    PA63      LOGICAL ERROR
01092A F4A8 96 51     A PA61   LDAA   INDATA+1 GET GIVEN MONTH FOR ENDING SPEED CALCULATION
01093A F4AA 81 02     A        CMPA   #$02      CHECK FOR FEBRUARY .
01094A F4AC 26 18 F4C6         BNE    PA64      NO, IT IS NOT FEBRUARY
01095A F4AE D6 52     A        LDAB   INDATA+2 GET DAY FOR ENDING SPEED CALCULATION
01096A F4B0 C1 29     A        CMPB   #$29      COMPARE WITH 29, THE EXTRA DAY ON LEAP YEAR
01097A F4B2 26 12 F4C6         BNE    PA64      NO, IT IS NOT THE 29TH
01098                      *
01099                      *       29TH OF FEBRUARY - DETERMINE WHETHER IT IS LEAP YEAR
01100                      *
01101A F4B4 4F                 CLRA             SET A=0
01102A F4B5 5F                 CLRB             SET B=0
01103A F4B6 8B 01     A PA62   ADDA   #1        A=A+1
01104A F4B8 19                 DAA              CONVERT TO BCD FORMAT
01105A F4B9 5C                 INCB             B=B+1
01106A F4BA 91 50     A        CMPA   INDATA    COMPARE YEAR WITH GIVEN YEAR FOR ENDING SPEED CALCULATION
01107A F4BC 26 F8 F4B6         BNE    PA62      NO - IT IS NOT
01108A F4BE C5 03     A        BITB   #%00000011 IS IT A LEAP YEAR
01109A F4C0 27 04 F4C6         BEQ    PA64      YES - LEAP YEAR, OKAY TO PROCESS
01110A F4C2 BD F651   A PA63   JSR    LOGER     PREPARE LOGICAL ERROR MESSAGE
01111A F4C5 39                 RTS              EXIT
01112                      *
01113                      *       EVERYTHING OKAY - PROCESS MESSAGE
01114                      *
01115A F4C6 96 6C     A PA64   LDAA   MFLAG     GET BYTE CONTAINING FLAGS
01116A F4C8 8A 02     A        ORAA   #%00000010 SET STOP SPEED CALCULATION FLAG
01117A F4CA 97 6C     A        STAA   MFLAG     SAVE UPDATED MFLAG
01118A F4CC DE 50     A        LDX    INDATA    GET YEAR AND MONTH FOR ENDING SPEED CALCULATION
01119A F4CE DF 69     A        STX    EDATE     SAVE THEM
01120A F4D0 96 52     A        LDAA   INDATA+2 GET DAY FOR ENDING SPEED CALCULATION
```

```
PAGE  016  PREANS  .SA:1  PREANS

01121A F4D2 97 6B    A       STAA   EDAY      SAVE IT
01122A F4D4 BD F5EB  A       JSR    RESCOM    GO TO PROCESS RESET CONTROL
01123                  *
01124                  *       PREPARE ANSWER
01125                  *
01126A F4D7 96 00    A       LDAA   PCSCAD    GET CONTROLER ADDRESS
01127A F4D9 8B C0    A       ADDA   #%11000000 ADD IN TRANSMISSION CODE
01128A F4DB 97 40    A       STAA   TCODE     SAVE IT
01129A F4DD CE 0008  A       LDX    #8        PUT 8 INTO X-REGISTER
01130A F4E0 DF 41    A       STX    MSGLEN    SAVE MESSAGE LENGTH
01131A F4E2 CE 0050  A       LDX    #INDATA   GET LOCATION OF INPUT BUFFER
01132A F4E5 DF 47    A       STX    DPOINT    SET DPOINT TO BEGINNING OF INPUT BUFFER
01133A F4E7 CE 0003  A       LDX    #3        PUT 3 INTO X-REGISTER
01134A F4EA DF 49    A       STX    DLNGTH    SET DLNGTH EQUAL TO THREE
01135A F4EC CE 0000  A       LDX    #0        CLEAR X-REGISTER
01136A F4EF DF 4B    A       STX    POINT2    CLEAR SECOND BUFFER POINTER
01137A F4F1 DF 4D    A       STX    LNGTH2    CLEAR LENGTH OF SECOND BUFFER
01138A F4F3 39               RTS              EXIT TO COMMUNICATIONS
```

```
01140                         *
01141                         *       CHANGE FIELD STATION MEMORY - MESSAGE 111
01142                         *
01143                         *
01144                         *       THIS MESSAGE WILL CHANGE CONTENTS OF RAM MEMORY UNLESS PROTECTED OR PART OF THE
01145                         *       SYSTEM STACK.
01146                         *    ***BODY OF MESSAGE
01147                         *       INPUT
01148                         *         BYTE             CONTENTS
01149                         *           1              STARTING ADDRESS OF CHANGE ( MSB )
01150                         *           2              STARTING ADDRESS OF CHANGE ( LSB )
01151                         *           3              NUMBER OF BYTES TO CHANGE ( MAX 16 )
01152                         *           4 - CHECKSUM   NEW MEMORY CONTENTS REQUESTED
01153                         *       OUTPUT
01154                         *         BYTE             CONTENTS
01155                         *           1              ( AS IN
01156                         *           2                INPUT
01157                         *           3                MESSAGE )
01158                         *           4 TO ERROR     STATE OF MEMORY AFTER CHANGE
01159                         *             CODE BYTE
01160                         *       THE FOLLOWING CONDITIONS WILL GENERATE A LOGICAL ERROR MESSAGE:
01161                         *       - INCORRECT BODY OF MESSAGE
01162                         *       - AREA TO BE CHANGED PARTIALLY OR COMPLETELY OUT OF RANGE OF RAM
01163                         *       - SECURITY CODE NOT CORRECT IN CORE & NOT PROVIDED IN THE MESSAGE OR INCORRECTLY
01164                         *         PROVIDED IN THE MESSAGE
01165                         *
01166                         *
01167                         *        CHECK FOR MINIMUM ONE BYTE OF DATA
01168                         *
01169A F4F4 CE 0052  A PA7     LDX    #INDATA+2 GET ADDRESS OF THIS BYTE
01170A F4F7 DF 3E    A         STX    TEMP01   SAVE IT
01171A F4F9 96 48    A         LDAA   DPOINT+1 LSB OF ADDRESS OF LAST CHARACTER RECEIVED
01172A F4FB 91 3F    A         CMPA   TEMP02   CHECK IF ENOUGH DATA
01173A F4FD 23 68 F567         BLS    PA70     NOT ENOUGH - PREPARE LOGICAL ERROR MESSAGE
01174A F4FF 96 48    A         LDAA   DPOINT+1 GET POINTER TO LAST CHARACTER (LSB)
01175A F501 CE 0052  A         LDX    #INDATA+2 GET ADDRESS OF NUMBER OF BYTES TO CHANGE
01176A F504 DF 3E    A         STX    TEMP01   SAVE IT
01177A F506 90 3F    A         SUBA   TEMP02   CALCULATE NUMBER OF BYTES OF CHANGES ACTUALLY RECEIVED
01178A F508 91 52    A         CMPA   INDATA+2 COMPARE WITH VALUE IN MESSAGE
01179A F50A 26 5B F567         BNE    PA70     NOT CORRECT - PREPARE LOGICAL ERROR MESSAGE
01180                         *
01181                         *       CALCULATE MAX ADDRESS REQUIRED AND COMPARE WITH RAM AVAILABLE
01182                         *
01183A F50C 96 52    A         LDAA   INDATA+2 GET LENGTH
01184A F50E 4A                 DECA            DECREMENT TO FIND ADDRESS OF THE LAST BYTE
01185A F50F 9B 51    A         ADDA   INDATA+1 ADD LSB OF STARTING ADDRESS
01186A F511 97 3F    A         STAA   TEMP02   SAVE IT
01187A F513 96 50    A         LDAA   INDATA+0 GET MSB OF STARTING ADDRESS
01188A F515 89 00    A         ADCA   #0       ADD IN CARRY BIT IF ANY
01189A F517 97 3E    A         STAA   TEMP01   SAVE IT
01190A F519 91 6E    A         CMPA   MRAM+0   COMPARE WITH RAM AVAILABLE
01191A F51B 22 4A F567         BHI    PA70     LOGICAL ERROR
01192A F51D 26 06 F525         BNE    PA7X     OKAY
01193A F51F 96 3F    A         LDAA   TEMP02   GET LSB OF MAX ADDRESS
01194A F521 91 6F    A         CMPA   MRAM+1   COMPARE WITH END OF RAM ADDRESS ( LSB )
01195A F523 22 42 F567         BHI    PA70     RAM NOT AVAILABLE - PREPARE LOGICAL ERROR MESSAGE
01196                         *
01197                         *       CHECK FOR CORRECT SECURITY CODE
```

```
PAGE  019  PREANS  .SA:1  PREANS

01198                    *
01199A F525 CE 0705  A PA7X   LDX    #$0705   GET CORRECT SECURITY CODE
01200A F528 9C 01    A        CPX    SECCOD   COMPARE WITH VALUE STORED
01201A F52A 27 3F F56B        BEQ    PA71     CORRECT VALUE - DO PROCESSING
01202                    *
01203                    *      CHECK FOR SECURITY CODE SUPPLIED IN THE MESSAGE
01204                    *
01205                    *
01206                    *       IS STARTING ADDRESS IN MESSAGE LESS THAN OR EQUAL TO SECURITY CODE ADDRESS
01207                    *
01208A F52C CE 0001  A        LDX    #SECCOD  GET SECURITY CODE ADDRESS
01209A F52F DF 3E    A        STX    TEMP01   SAVE IT
01210A F531 96 50    A        LDAA   INDATA   GET STARTING ADDRESS OF CHANGE (MSB)
01211A F533 91 3E    A        CMPA   TEMP01   COMPARE MSB'S OF ADDRESS
01212A F535 22 30 F567        BHI    PA70     LOGICAL ERROR
01213A F537 26 06 F53F        BNE    PA7Y     OKAY - IE. LESS THAN
01214A F539 96 51    A        LDAA   INDATA+1 CHECK LSB
01215A F53B 91 3F    A        CMPA   TEMP02
01216A F53D 22 28 F567        BHI    PA70     LOGICAL ERROR MESSAGE
01217A F53F 96 3F    A PA7Y   LDAA   TEMP02   GET LSB OF ADDRESS
01218A F541 90 51    A        SUBA   INDATA+1 SUBTRACT LSB OF DATA ADDRESS TO GET DISPLACEMENT INTO DATA IN BUFFER
01219A F543 D6 52    A        LDAB   INDATA+2 # OF BYTES TO CHANGE
01220A F545 C0 02    A        SUBB   #2       SUBTRACT LENGTH OF SECURITY CODE
01221A F547 25 1E F567        BCS    PA70     ONLY ONE BYTE - ERROR
01222A F549 11                CBA             FIND IF SECURITY CODE IS INDEED IN THE NEW DATA PROVIDED
01223A F54A 22 1B F567        BHI    PA70     NO SECURITY CODE
01224                    *
01225                    *      SECURITY CODE IS SUPPLIED IN THE MESSAGE - FIND IF VALUE CORRECT
01226                    *
01227A F54C CE 0050  A        LDX    #INDATA  BEGINNING OF BUFFER ADDRESS
01228A F54F DF 3E    A        STX    TEMP01   SAVE IN TEMPORY LOCATION
01229A F551 5F                CLRB            CLEAR B FOR 2 BYTE ADD
01230A F552 8B 03    A        ADDA   #3       DISPLACEMENT + 3
01231A F554 9B 3F    A        ADDA   TEMP02   ADD LSB OF ADDRESS
01232A F556 97 3F    A        STAA   TEMP02   SAVE IT
01233A F558 D9 3E    A        ADCB   TEMP01   CORRECT MSB OF ADDRESS
01234A F55A D7 3E    A        STAB   TEMP01   SAVE IT
01235A F55C DE 3E    A        LDX    TEMP01   NOW X-REGISTER CONTAINS ADDRESS OF NEW SECURITY CODE
01236                    *                    SUPPLIED IN THE MESSAGE
01237A F55E EE 00    A        LDX    0,X      LOAD VALUE OF SECURITY CODE SUPPLIED IN THE MESSAGE
01238A F560 8C 0705  A        CPX    #$0705   CHECK IF VALUE CORRECT
01239A F563 26 02 F567        BNE    PA70     NOT CORRECT - ERROR
01240A F565 20 04 F56B        BRA    PA71     PROCESS MESSAGE
01241                    *
01242                    *      ERROR IN THE MESSAGE
01243                    *
01244A F567 BD F651  A PA70   JSR    LOGER    PREPARE LOGICAL ERROR MESSAGE
01245A F56A 39                RTS             EXIT
01246                    *
01247                    *      PROCESS MESSAGE
01248                    *
01249A F56B 7F 0065  A PA71   CLR    CTIMS3   CLEAR SECURITY CODE TIMER
01250A F56E 96 6C    A        LDAA   MFLAG    GET FLAG BITS
01251A F570 8A 10    A        ORAA   #%00010000 SET FLAG
01252A F572 97 6C    A        STAA   MFLAG    SAVE UPDATED FLAG BYTE
01253                    *
01254                    *      IMPLEMENT RAM CHANGES
01255                    *
```

```
PAGE  019  PREANS  .SA:1  PREANS

01256A F574 DE 50     A        LDX    INDATA   GET INITIAL RAM ADDRESS
01257A F576 DF 3E     A        STX    TEMP01   SAVE IT
01258A F578 CE 0053   A        LDX    #INDATA+3 POINT TO FIRST BYTE OF NEW DATA
01259A F57B DF 47     A        STX    DPOINT   SAVE IT
01260A F57D 96 52     A        LDAA   INDATA+2 GET # OF BYTES TO BE CHANGED
01261A F57F DE 47     A PA72   LDX    DPOINT   GET ADDRESS OF NEXT NEW BYTE
01262A F581 E6 00     A        LDAB   0,X      GET NEW DATA
01263A F583 DE 3E     A        LDX    TEMP01   GET RAM ADDRESS
01264                 *
01265                 *       CHECK FOR PROTECTED LOCATIONS
01266                 *
01267A F585 8C 0033   A        CPX    #TMFLG1  CHECK FOR PROTECTED LOCATION
01268A F588 27 2F F5B9         BEQ    PA73     YES
01269A F58A 8C 0034   A        CPX    #TMFLG2  ANOTHER PROTECTED LOCATION
01270A F58D 27 2A F5B9         BEQ    PA73     YES
01271A F58F 8C 006E   A        CPX    #MRAM    AGAIN
01272A F592 27 25 F5B9         BEQ    PA73     YES
01273A F594 8C 006F   A        CPX    #MRAM+1  AGAIN
01274A F597 27 20 F5B9         BEQ    PA73     YES
01275A F599 8C 0035   A        CPX    #EXFLG   AGAIN
01276A F59C 27 1B F5B9         BEQ    PA73     YES
01277A F59E 8C 0038   A        CPX    #EXINDX  AGAIN
01278A F5A1 27 16 F5B9         BEQ    PA73     YES
01279A F5A3 8C 0039   A        CPX    #EXINDX+1 AGAIN
01280A F5A6 27 11 F5B9         BEQ    PA73     YES
01281A F5A8 8C 003A   A        CPX    #SBRINX  AGAIN
01282A F5AB 27 0C F5B9         BEQ    PA73
01283A F5AD 8C 003B   A        CPX    #SBRINX+1 AGAIN
01284A F5B0 27 07 F5B9         BEQ    PA73     YES
01285A F5B2 8C 0020   A        CPX    #KYSTAT  AGAIN
01286A F5B5 27 02 F5B9         BEQ    PA73     YES
01287                 *
01288                 *       OTHER PROTECTED LOCATIONS TO BE IDENTIFIED LATER
01289                 *
01290A F5B7 E7 00     A        STAB   0,X      STORE NEW DATA INTO RAM
01291A F5B9 4A          PA73   DECA            CALCULATE # OF BYTES STILL TO BE CHANGED
01292A F5BA 27 0A F5C6         BEQ    PA74     ALL DONE
01293A F5BC 08                 INX             ADVANCE RAM ADDRESS
01294A F5BD DF 3E     A        STX    TEMP01   SAVE IT
01295A F5BF DE 47     A        LDX    DPOINT   GET ADDRESS OF NEW DATA
01296A F5C1 08                 INX             POINT TO NEXT BYTE
01297A F5C2 DF 47     A        STX    DPOINT   SAVE IT
01298A F5C4 20 B9 F57F         BRA    PA72     DO NEXT BYTE
01299                 *
01300                 *       ALL CHANGES DONE - PREPARE ANSWER MESSAGE
01301                 *
01302A F5C6 96 00     A PA74   LDAA   PCSCAD   GET CONTROLLER ADDRESS
01303A F5C8 8B E0     A        ADDA   #%11100000 ADD IN MESSAGE CODE
01304A F5CA 97 40     A        STAA   TCODE    SAVE IT
01305A F5CC 96 52     A        LDAA   INFATA+2 NUMBER OF BYTES TO CHANGE
01306A F5CE 8B 08     A        ADDA   #8       CALCULATE LENGTH OF MESSAGE
01307A F5D0 7F 0041   A        CLR    MSGLEN   MSB OF LENGTH
01308A F5D3 97 42     A        STAA   MSGLEN+1 LSB OF LENGTH
01309A F5D5 CE 0050   A        LDX    #INDATA  ADDRESS OF FIRST BUFFER
01310A F5D8 DF 47     A        STX    DPOINT   SAVE IT
01311A F5DA CE 0003   A        LDX    #3       LENGTH OF FIRST BUFFER
01312A F5DD DF 49     A        STX    DLNGTH   SAVE IT
01313A F5DF DE 50     A        LDX    INDATA   GET ADDRESS OF RAM JUST CHANGED
```

```
PAGE  020  PREANS  .SA:1  PREANS

01314A F5E1 DF 4B    A        STX    POINT2   SAVE IT
01315A F5E3 96 52    A        LDAA   INDATA+2 GET LENGTH OF SECOND BUFFER
01316A F5E5 7F 004D  A    .   CLR    LNGTH2   MSB OF LENGTH
01317A F5E8 97 4E    A        STAA   LNGTH2+1 LSB OF LENGTH
01318A F5EA 39                RTS             EXIT
```

```
PAGE  021  PREANS  .SA:1  PREANS

01320                        *
01321                        *       SUBROUTINE RESCON - RESET DETECTOR ERROR FLAG AND TIME DOWN
01322                        *       COUNT AS APPLICABLE
01323                        *
01324                        *       RESET CONTROL ERROR BYTE
01325                        *       DETECTOR LOCATION BITS
01326                        *         BIT              MEANING
01327                        *          0               DIRECTION---------------         DIR.
01328                        *                                          --- 0 -----  1
01329                        *                                          --- 1 -----  2
01330                        *          1               UPSTREAM/DOWNSTREAM-----
01331                        *                                          --- 0 -----  UP
01332                        *                                          --- 1 -----  DOWN
01333                        *          2  >
01334                        *          3  > --LANES  ( 000 ---> 111 )
01335                        *          4  >
01336                        *       ACTION BITS
01337                        *          5               CLEAR DETECTOR ERROR --- 0 ----- DO NOTHING
01338                        *                                          --- 1 ----- CLEAR THE CORRESPONDING DETECTOR
01339                        *          6               CLEAR TIME DOWN COUNT IF SET
01340                        *          7               RESERVED FOR FUTURE USE
01341                        *
01342A F5EB 96 53      A RESCON LDAA   INDATA+3 GET RESET CONTROL BYTE FROM MESSAGE
01343A F5ED 85 40      A        BITA   #%01000000 CHECK FOR CLEAR TIME COUNT BIT
01344A F5EF 27 09 F5FA          BEQ    RES1     NO REQUEST TO CLEAR TIME DOWN COUNT
01345                        *
01346                        *      CLEAR TIME DOWN COUNT ETC.
01347                        *
01348A F5F1 D6 44      A        LDAB   ERCODE   GET ERROR CODE BYTE
01349A F5F3 C4 FB      A        ANDB   #%11111011 RESET TIME-OF-DAY CLOCK ERROR BIT
01350A F5F5 D7 44      A        STAB   ERCODE   SAVE IT
01351A F5F7 7F 0036    A        CLR    DWNCNT   CLEAR POWER FAILURE COUNTER
01352                        *
01353                        *      CHECK FOR AND CLEAR DETECTOR ERROR AS APPLICABLE
01354                        *
01355A F5FA 85 20      A RES1   BITA   #%00100000 CHECK CLEAR DETECTOR ERROR BIT
01356A F5FC 26 01 F5FF          BNE    RES2     BIT SET - PROCESS
01357A F5FE 39                  RTS             NO REQUEST - EXIT
01358                        *
01359                        *      THERE IS A REQUEST TO CLEAR DETECTOR ERROR
01360                        *
01361A F5FF C6 FE      A RES2   LDAB   #%11111110 PREPARE LANE RESETTING BYTE
01362A F601 84 1C      A        ANDA   #%00011100 CLEAR UNNESESSARY INFO
01363A F603 44                  LSRA            (RIGHT JUSTIFY
01364A F604 44                  LSRA            LANE NUMBER)
01365A F605 0D                  SEC             SET CARRY BIT FOR SHIFTING
01366A F606 27 04 F60C RES3     BEQ    RES4     NO MORE SHIFTING REQUIRED
01367A F608 59                  ROLB            MOVE LEFT
01368A F609 4A                  DECA            SUBTRACT FROM LANE NUMBER
01369A F60A 20 FA F606          BRA    RES3     CHECK IF ANOTHER SHIFT REQUIRED
01370                        *
01371                        *      NOW LANE RESETTING BYTE IS AVAILABLE IN REG B
01372                        *
01373A F60C D7 3E      A RES4   STAB   TEMP01   SAVE RESETTING BYTE
01374A F60E 96 53      A        LDAA   INDATA+3 GET RESET CONTROL BYTE AGAIN
01375A F610 85 01      A        BITA   #%00000001 CHECK DIRECTION
01376A F612 26 14 F628          BNE    RES6     BRANCH IF DIRECTION 2
01377                        *
```

```
PAGE  022  PREANS  .SA:1  PREANS

01378                          *        DIRECTION 1
01379                          *
01380A F614 85 02     A          BITA   #%00000010 CHECK FOR UPSTREAM/DOWNSTREAM
01381A F616 26 08 F620           BNE    RES5      BRANCH IF DOWNSTREAM
01382A F618 D6 8C     A          LDAB   DETER1    GET PROPER DETECTOR ERROR CODE BYTE
01383A F61A D4 3E     A          ANDB   TEMP01    RESET APPROPRIATE BIT
01384A F61C D7 8C     A          STAB   DETER1    STORE BACK
01385A F61E 20 1A F63A           BRA    RES8      CHECK FOR OTHER ERRORS
01386A F620 D6 8D     A RES5     LDAB   DETER2    GET DETECTOR ERROR BYTE
01387A F622 D4 3E     A          ANDB   TEMP01    RESET
01388A F624 D7 8D     A          STAB   DETER2    STORE BACK
01389A F626 20 12 F63A           BRA    RES8      CHECK FOR OTHER ERRORS
01390                          *
01391                          *        DIRECTION 2
01392                          *
01393A F628 85 02     A RES6     BITA   #%00000010 CHECK FOR UPSTREAM/DOWNSTREAM
01394A F62A 26 08 F634           BNE    RES7      BRANCH IF DOWNSTREAM
01395A F62C D6 8E     A          LDAB   DETER3    GET DETECTOR ERROR BYTE
01396A F62E D4 3E     A          ANDB   TEMP01    RESET APPROPRIATE BIT
01397A F630 D7 8E     A          STAB   DETER3    STORE BACK
01398A F632 20 06 F63A           BRA    RES8      CHECK FOR OTHER ERRORS
01399A F634 D6 8F     A RES7     LDAB   DETER4    GET ERROR BYTE
01400A F636 D4 3E     A          ANDB   TEMP01    RESET
01401A F638 D7 8F     A          STAB   DETER4    STORE BACK
01402                          *
01403                          *        DETECTOR ERROR BIT RESET AS REQUIRED - CHECK FOR OTHER ERRORS
01404                          *
01405A F63A 96 8C     A RES8     LDAA   DETER1    CHECK FIRST BYTE FOR ERRORS
01406A F63C 26 12 F650           BNE    RES9      BRANCH IF ERROR FOUND
01407A F63E 96 8D     A          LDAA   DETER2    CHECK SECOND BYTE.
01408A F640 26 0E F650           BNE    RES9      BRANCH IF ERROR FOUND
01409A F642 96 8E     A          LDAA   DETER3    CHECK THIRD BYTE
01410A F644 26 0A F650           BNE    RES9      BRANCH IF ERROR FOUND
01411A F646 96 8F     A          LDAA   DETER4    CHECK LAST BYTE
01412A F648 26 06 F650           BNE    RES9      BRANCH IF ERROR FOUND
01413                          *
01414                          *        NO DETECTOR ERRORS LEFT
01415                          *
01416A F64A 96 44     A          LDAA   ERCODE    GET ERROR CODE BYTE
01417A F64C 84 FE     A          ANDA   #%11111110 RESET ERROR BIT
01418A F64E 97 44     A          STAA   ERCODE    STORE BACK
01419A F650 39          RES9     RTS              EXIT
```

```
PAGE  023  PREANS  .SA:1  PREANS

01421                   *
01422                   *     SUBROUTINE LOGER - PREPARE LOGICAL ERROR MESSAGE
01423                   *
01424A F651 96 00     A LOGER  LDAA   PCSCAD   GET CONTROLLER'S ADDRESS
01425A F653 97 40     A        STAA   TCODE    MESSAGE 000, SAVE IN TCODE
01426A F655 CE 0005   A        LDX    #5       LENGTH OF MESSAGE =5
01427A F658 DF 41     A        STX    MSGLEN   STORE TO LENGTH FIELD
01428A F65A CE 0000   A        LDX    #0       CLEAR X-REGISTER
01429A F65D DF 47     A        STX    DPOINT   (CLEAR
01430A F65F DF 49     A        STX    DLNGTH   POINTERS
01431A F661 DF 4B     A        STX    POINT2   AND LENGTHS
01432A F663 DF 4D     A        STX    LNGTH2   FOR BOTH OUTPUT BUFFERS)
01433A F665 39                 RTS             EXIT
01434                   *
01435                   *     END OF PROGRAM
01436                   *
01437                          END
TOTAL ERRORS 00000--00000
```

```
PAGE  001  EQUATE  .SA:1  TIMERS

00498                        OPT    L
00499                        NAM    TIMERS
00500                 *
00501                 *          COMMUNICATION TIMER ROUTINES
00502                 *
00503                 *
00504                 *             ****************************************************
00505                 *             *                                                  *
00506                 *             *   PROGRAMMER:   W.J. WIERCIENSKI                 *
00507                 *             *                                                  *
00508                 *             *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS  *
00509                 *             *   TRANSPORTATION TECHNOLOGY AND ENERGY           *
00510                 *             *   1201 WILSON AVENUE                             *
00511                 *             *   DOWNSVIEW, ONTARIO  M3M 1J8                    *
00512                 *             *                                                  *
00513                 *             *   TELEPHONE (416) 248-7141                       *
00514                 *             *                                                  *
00515                 *             *   NOVEMBER  1982 AD                              *
00516                 *             *                                                  *
00517                 *             ****************************************************
00518                 *
00519                 *
00520                 *          SUBROUTINE TIMER1
00521                 *
00522                 *          THIS ROUTINE IS EXECUTED EVERY 1/10TH SECOND. ITS PURPOSE IS TO INCREMENT
00523                 *          AND EXAMINE THE COMMUNICATION TIMERS 1&2. TIMER 1 IS USED TO RESOLVE THE
00524                 *          DEADLOCK SITUATION WHILE RECEIVING, IE. IF THERE IS AN UNDUE DELAY IN THE
00525                 *          RECEIVING ANYWHERE IN THE MESSAGE ( CAUSED BY THE CENTRAL COMPUTER ).
00526                 *          TIMER 2 TIMES THE GAP BETWEEN THE CHARACTERS TRANSMITTED & BREAKS THE
00527                 *          DEADLOCK IF THERE IS A DELAY DUE TO LOSS OF THE DATA CARRIER. ( THE DATA
00528                 *          CARRIER LOSS INTERRUPTS ARE INHIBITED WHILE TRANSMITTING ).
00529                 *
00530                 *          THERE ARE THREE POSSIBLE STATES FOR THE COMMUNICATIONS ROUTINE TO BE IN:
00531                 *          STATE 1 - RBITS=1 > WAITING NEW MESSAGE AFTER
00532                 *                    XBITS=0 > -POWER UP
00533                 *                              -PREVIOUS MESSAGE
00534                 *                              -DATA CARRIER LOSS IN THE MIDDLE OF MESSAGE
00535                 *                    STATE 1 DOES NOT REQUIRE ANY ACTION TO BE TAKEN
00536                 *          STATE 2 - RBITS>1 > IN THE RECEIVING STATE
00537                 *                    XBITS=0 > - ANSWER READY ( GOOD ANSWER OR ERROR MESSAGE )
00538                 *                              - NO ANSWER PREPARED AS YET - PREPARE ERROR MESSAGE
00539                 *                    LINE OKAY, SEND MESSAGE BACK
00540                 *          STATE 3 - RBITS=0 > IN TRANSMITTING STATE
00541                 *                    XBITS>0 > -FAILURE MUST BE DUE TO CARRIER LOSS (MASKED)
00542                 *                    NOT POSSIBLE TO SEND MESSAGE AS THE LINE IS DOWN - TURN BACK TO RECEIVING
00543                 *
```

```
PAGE  002  TIMERS  .SA:1  TIMERS

00545A F670                       ORG    TIMER1
00546A F670 96 45          A      LDAA   RBITS    GET RECEIVING STATE INDICATOR
00547A F672 81 01          A      CMPA   #1       IS IT IN STATE 1
00548A F674 26 01 F677            BNE    TS1      NO - MUST BE IN STATE 2, OR 3
00549A F676 39                    RTS             STATE 1 - DO NOTHING, EXIT
00550A F677 22 02 F67B       TS1  BHI    TS2      STATE 2
00551A F679 20 26 F6A1            BRA    TS5      MUST BE STATE 3
00552                        *
00553                        *         IT IS IN STATE 2; RECEIVING
00554                        *
00555A F67B 96 63          A TS2  LDAA   CTIME1   GET 1/10TH SECOND RECEIVING TIMER
00556A F67D 4C                    INCA            INCREMENT BY 1/10TH SECOND
00557A F67E 97 63          A      STAA   CTIME1   STORE FOR NEXT TIME
00558A F680 81 1E          A      CMPA   #30      HAS MAXIMUM RECEIVING DURATION BEEN REACHED
00559                        *                    (LONGEST MESSAGE SHOULD BE COMPLETED IN ABOUT 1 SECOND, BUT DUE TO
00560                        *                    POSSIBLE DELAYS ON THE NOVA SIDE, THE MAXIMUM TIME IS SET AT THREE SECONDS
00561A F682 27 01 F685            BEQ    TS3      YES, THE TIMER HAS EXPIRED
00562A F684 39                    RTS             NO, MAXIMUM RECEIVE DURATION NOT YET EXPIRED
00563                        *
00564                        *         RECEIVING DURATION TIMER HAS EXPIRED, IS THERE A PROPER RETURN MESSAGE READY
00565                        *
00566A F685 96 45          A TS3  LDAA   RBITS    GET RECEIVE STATE INDICATOR
00567A F687 81 04          A      CMPA   #4       HAS A RETURN MESSAGE BEEN PREPARED
00568A F689 22 12 F69D            BHI    TS4      YES, GO TO TRANSMIT THE FIRST CHARACTER
00569                        *
00570                        *         NO MESSAGE TO SEND - PREPARE A NEGATIVE ACKNOWLEDGEMENT MESSAGE
00571                        *
00572A F68B 96 00          A      LDAA   PCSCAD   GET FIELD STATION ADDRESS
00573A F68D 8B 40          A      ADDA   #%01000000 ADD CODE FOR A NEGATIVE ACKNOWLEDGEMENT MESSAGE
00574A F68F 97 40          A      STAA   TCODE    SAVE IT FOR TRANSMITTING
00575A F691 CE 0005        A      LDX    #5       BYTE COUNT TO CHECKSUM=5
00576A F694 DF 41          A      STX    MSGLEN   LENGTH OF NEGATIVE ACKNOWLEDGEMENT MESSAGE=5
00577A F696 CE 0000        A      LDX    #0       CLEAR X-REGISTER
00578A F699 DF 47          A      STX    DPOINT   ( CLEAR POINTERS
00579A F69B DF 49          A      STX    DLNGTH   FOR OUTPUT BUFFERS )
00580                        *
00581                        *         TRANSMIT FIRST CHARACTER, EITHER PROPER ANSWER OR NEGATIVE ACKNOWLEDGEMENT
00582                        *
00583A F69D BD F12F        A TS4  JSR    TFIRST   TRANSMIT FIRST CHARACTER
00584A F6A0 39                    RTS             EXIT
00585                        *
00586                        *         IT IS IN STATE 3; TRANSMITTING
00587                        *
00588A F6A1 96 64          A TS5  LDAA   CTIME2   GET TRANSMITTING GAP TIMER (2)
00589A F6A3 4C                    INCA            INCREMENT BY 1/10TH SECOND
00590A F6A4 97 64          A      STAA   CTIME2   STORE BACK
00591A F6A6 81 03          A      CMPA   #3       HAS MAXIMUM GAP TIME BEEN REACHED ( 3/10TH SECOND MAXIMUM
00592                        *                    USED AS START OF XMITTING COULD HAVE HAPPENED JUST BEFORE THIS S/R )
00593A F6A8 27 01 F6AB            BEQ    TS6      YES, THERE MUST BE A CARRIER LOSS
00594A F6AA 39                    RTS             NO, GAP TIMER NOT YET EXPIRED
00595                        *
00596                        *         TRANSMITTING GAP TIMER HAS EXPIRED, GO BACK TO RECEIVING STATE
00597                        *
00598A F6AB 86 9A          A TS6  LDAA   #%10011010 RECEIVER ON, TRANSMITTER OFF
00599                        *                    ( 8 DATA BITS, EVEN PARITY, 1 STOP BIT, 300 BAUD )
00600A F6AD B7 6000        A      STAA   ACIACS   RESET RECEIVING STATE
00601A F6B0 7F 0046        A      CLR    XBITS    CLEAR TRANSMITTING INDICATOR
00602A F6B3 86 01          A      LDAA   #1
```

```
PAGE  003  TIMERS  .SA:1  TIMERS

00603A F6B5 97 45    A         STAA   RBITS    RECEIVING INDICATOR SET TO START RECEIVING
00604A F6B7 39                 RTS             EXIT
```

```
PAGE  004  TIMERS  .SA:1  TIMERS

00606                    *
00607                    *        SUBROUTINE TIMER2
00608                    *
00609                    *        THIS ROUTINE IMPLEMENTS A 255 MINUTE TIMER. IT IS USED TO WIPE OUT THE
00610                    *        SECURITY CODE AFTER IT HAS BEEN SET BY THE COMMUNICATION MESSAGE.
00611                    *
00612A F6C0                    ORG    TIMER2
00613                    *
00614                    *        IS THE SECURITY CODE TIMER FLAG SET?
00615                    *
00616A F6C0 96 6C     A          LDAA   MFLAG     GET BYTE CONTAINING TIMER FLAG
00617A F6C2 85 10     A          BITA   #%00010000 IS THE FLAG SET
00618A F6C4 26 01 F6C7           BNE    TM1       YES IT IS SET
00619A F6C6 39                   RTS              NO - EXIT
00620                    *
00621                    *        THE TIMER IS RUNNING
00622                    *
00623A F6C7 96 65     A TM1      LDAA   CTIME3    GET TIMER BYTE TO BE INCREMENTED EVERY MINUTE
00624A F6C9 4C                   INCA             INCREMENT BY 1 MINUTE
00625A F6CA 97 65     A          STAA   CTIME3    SAVE IT
00626A F6CC 81 FF     A          CMPA   #255      HAS THE TIMER EXPIRED
00627A F6CE 27 01 F6D1           BEQ    TM2       YES THE TIMER HAS EXPIRED
00628A F6D0 39                   RTS              NO - NOT YET
00629                    *
00630                    *        THE TIMER HAS EXPIRED CLEAR THE SECURITY CODE
00631                    *
00632A F6D1 CE 0000   A TM2      LDX    #0        ( CLEAR THE
00633A F6D4 DF 01     A          STX    SECCOD    SECURITY CODE )
00534A F6D6 96 6C     A          LDAA   MFLAG     GET THE BYTE CONTAINING THE SECURITY CODE FLAG
00635A F6D8 84 EF     A          ANDA   #%11101111 RESET THE FLAG
00636A F6DA 97 6C     A          STAA   MFLAG     SAVE RESULT
00637A F6DC 39                   RTS              EXIT
00638                            END
TOTAL ERRORS 00000--00000
```

```
PAGE  001  EQUATE  .SA:1  MASINT

                      OPT     L
                      NAM     MASINT
                 *
                 *
                 *        MASTER INITIALIZATION PROGRAM
                 *
                 *
                 *            *************************************************************
                 *            *                                                           *
                 *            *    PROGRAMMER:    W.J. WIERCIENSKI                        *
                 *            *                                                           *
                 *            *    MINISTRY OF TRANSPORTATION AND COMMUNICATIONS          *
                 *            *    TRANSPORTATION TECHNOLOGY AND ENERGY                   *
                 *            *    1201 WILSON AVENUE                                     *
                 *            *    DOWNSVIEW, ONTARIO  M3M 1J8                            *
                 *            *                                                           *
                 *            *    TELEPHONE (416) 248-7141                               *
                 *            *                                                           *
                 *            *    NOVEMBER  1982 AD                                      *
                 *            *                                                           *
                 *            *************************************************************
                 *
                 *
                 *        THIS PROGRAM INITIALIZES SYSTEM & APPLICATION
                 *        VARIABLES AS REQUIRED
                 *        IT IS STARTED EVERY SECOND AND EXECUTED IF
                 *        MASTER FLAG IS SET
                 *
```

```
PAGE  002  MASINT  .SA:1  MASINT

00527A F6F0                       ORG    MASINT
00528A F6F0 96 6D     A           LDAA   MBYTE    GET MASTER INITIALIZATION FLAG
00529A F6F2 27 48 F73C            BEQ    END      SKIP IF FLAG ZERO
00530A F6F4 BD D000  A            JSR    INIT     INITIALIZE APPLICATION VARIABLES
00531A F6F7 7F 006D  A            CLR    MBYTE    CLEAR FLAG
00532                     *
00533                     *          DELETE ERRORS
00534                     *
00535A F6FA 7F 0087  A            CLR    LFYEAR   CLEAR DATE OF LAST POWER FAILURE
00536A F6FD 7F 0088  A            CLR    LFMNTH   * * *
00537A F700 7F 0089  A            CLR    LFDAY    * * *
00538A F703 7F 008A  A            CLR    LFHR     CLEAR TIME OF LAST POWER FAILURE
00539A F706 7F 008B  A            CLR    LFMIN    * * *
00540A F709 7F 0036  A            CLR    DWNCNT   CLEAR POWER DOWN COUNTER
00541A F70C 7F 0037  A            CLR    ALARMS   RESET ALARMS
00542A F70F 7F 008C  A            CLR    DETER1   CLEAR
00543A F712 7F 008D  A            CLR    DETER2   DETECTOR
00544A F715 7F 008E  A            CLR    DETER3   ERROR
00545A F718 7F 008F  A            CLR    DETER4   WORDS
00546A F71B 7F 004F  A            CLR    CERROR   CLEAR COMMUNICATIONS ERROR WORD
00547A F71E 96 44     A           LDAA   ERCODE   GET ERROR CODE WORD
00548A F720 84 FB     A           ANDA   #%11111011 RESET CLOCK ERROR BIT
00549A F722 97 44     A           STAA   ERCODE   STORE BACK
00550                     *
00551                     *          RESET START AND END SPEED CALCULATION VARIABLES
00552                     *
00553A F724 96 6C     A           LDAA   MFLAG    GET FLAGS
00554A F726 84 FC     A           ANDA   #%11111100 RESET BITS 0 AND 1 -   ART/END SPEED CALCULATIONS
00555A F728 97 6C     A           STAA   MFLAG    SAVE FLAGS
00556A F72A 7F 0066  A            CLR    SYEAR    CLEAR START DATE
00557A F72D 7F 0067  A            CLR    SMONTH
00558A F730 7F 0068  A            CLR    SDAY
00559A F733 7F 0069  A            CLR    EYEAR    CLEAR END DATE
00560A F736 7F 006A  A            CLR    EMONTH
00561A F739 7F 006B  A            CLR    EDAY
00562A F73C 39            END     RTS             FINISH
00563                             END
TOTAL ERRORS 00000--00000
```

```
PAGE  001  EQUATE  .SA:1

00498                        OPT    L
00499                  *
00500                  *       ROUTINES TO PERFORM THE RAM TEST
00501                  *
00502                  *
00503                  *           ***************************************************
00504                  *           *                                                 *
00505                  *           *   PROGRAMMER:   W.J. WIERCIENSKI                *
00506                  *           *                                                 *
00507                  *           *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS *
00508                  *           *   TRANSPORTATION TECHNOLOGY AND ENERGY          *
00509                  *           *   1201 WILSON AVENUE                            *
00510                  *           *   DOWNSVIEW, ONTARIO  M3M 1J8                   *
00511                  *           *                                                 *
00512                  *           *   TELEPHONE (416) 248-7141                      *
00513                  *           *                                                 *
00514                  *           *   NOVEMBER  1982 AD                             *
00515                  *           *                                                 *
00516                  *           ***************************************************
00517                  *
00518                  *
00519                  *       THE FOLLOWING ROUTINES ARE USED TO PERFORM A RAM TEST
00520                  *       TO DETECT ANY BAD RAM CHIPS. THE RAM TEST IS INITIATED AT
00521                  *       POWER UP TIME BY POWRON OR IS STARTED EVERY 24 HOURS BY
00522                  *       CALPRO AFTER THE PROM CHECKSUM HAS BEEN PERFORMED.
00523                  *       THE RAM TEST PROCEEDS AT 1 BYTE EVERY 1/60 SEC., EXCEPT AT
00524                  *       POWER UP, WHEN THE WHOLE TEST IS PERFORMED AT ONCE. ANY
00525                  *       RAM ERRORS WILL CAUSE THE APPROPRIATE BITS OF THE RAM ERROR
00526                  *       FLAG RAMERR TO BE SET.
00527                  *
```

```
PAGE  002  RAMCHK  .SA:1

00529A F750                     ORG    RAM60
00530                   *
00531                   *          THIS ROUTINE WILL TEST 1 BYTE OF RAM MEMORY UNLESS A
00532                   *          RAM TEST HAS ALREADY BEEN COMPLETED FOR TODAY
00533                   *
00534A F750 96 6C    A          LDAA   MFLAG     GET MASTER FLAGS
00535A F752 85 40    A          BITA   #%01000000 HAVE RAM TESTS ALREADY BEEN DONE TODAY?
00536A F754 27 03 F759          BEQ    L1        YES -
00537A F756 BD F760  A          JSR    RAMCHK    NO - CHECK 1 BYTE OF RAM
00538A F759 39          L1      RTS
```

```
PAGE  003  RAMCHK  .SA:1

00540A F760                       ORG    RAMCHK
00541                       *
00542                       *        PERFORM A RAM TEST AND INDICATE ANY ERRORS FOR A 2K RAM CHIP
00543                       *        - IF BIT 6 OF MFLAG IS 0, RAMCHK WAS CALLED FROM POWRON, SO
00544                       *          THE ENTIRE RAM CHECK IS TO BE PERFORMED NOW
00545                       *        - IF BIT 6 OF MFLAG IS 1, RAMCHK WAS CALLED BY RAM60, SO
00546                       *          ONLY 1 BYTE OF RAM IS TO BE TESTED
00547                       *
00548A F760 DE 84     A           LDX    PADDR    GET ADDRESS OF RAM TO TEST
00549                       *
00550                       *        TEST 1 BYTE OF RAM - EXERCISE EACH BIT
00551                       *
00552A F762 4F              R0    CLRA            PREPARE REG. A FOR SHIFTING
00553A F763 0D                    SEC             PREPARE CARRY BIT FOR SHIFTING
00554A F764 0F                    SEI             DISABLE INTERRUPTS (SINCE WE ARE CHANGING MEMORY) **********
00555A F765 E6 00     A           LDAB   0,X      SAVE DATA IN REG. B (WE WILL PUT IT BACK LATER)
00556A F767 49              R1    ROLA            EXERCISE NEXT BIT
00557A F768 A7 00     A           STAA   0,X      SAVE IT
00558A F76A A1 00     A           CMPA   0,X      SAVED CORRECTLY?
00559A F76C 26 10 F77E            BNE    ERR      NO -RAM ERROR
00560A F76E 4D                    TSTA            FINISHED YET? (ALSO CLEARS CARRY FOR SHIFTING)
00561A F76F 26 F6 F767            BNE    R1       NO -
00562                       *       NOW TRY ALL BITS AT ONCE
00563A F771 86 FF     A           LDAA   #$FF     TEST ALL BITS
00564A F773 A7 00     A           STAA   0,X      SAVE IT
00565A F775 A1 00     A           CMPA   0,X      SAVED CORRECTLY?
00566A F777 26 05 F77E            BNE    ERR      NO -
00567                       *
00568                       *        THIS BYTE IS OK
00569                       *
00570A F779 E7 00     A           STAB   0,X      PUT MEMORY BACK THE WAY WE FOUND IT
00571A F77B 0E                    CLI             ENABLE INTERRUPTS **********
00572A F77C 20 1F F79D            BRA    R2
00573                       *
00574                       *        RAM ERROR - SET FLAG FOR THIS CHIP
00575                       *
00576A F77E E7 00     A ERR       STAB   0,X      PUT DATA BACK
00577A F780 0E                    CLI             ENABLE INTERRUPTS **********
00578A F781 96 82     A           LDAA   PCHECK   GET CHIP BIT INDICATOR
00579A F783 9A 2E     A           ORAA   RAMERR   SET APPROPRIATE RAM ERROR BIT
00580A F785 97 2E     A           STAA   RAMERR   SAVE IT
00581A F787 96 83     A           LDAA   PCHECK+1 REPEAT WITH SECOND BYTE (FOR CHIPS 9 AND 10)
00582A F789 9A 2F     A           ORAA   RAMERR+1
00583A F78B 97 2F     A           STAA   RAMERR+1
00584                       *
00585                       *        SET ADDRESS TO END OF THIS 2K CHIP
00586                       *        (ONE ERROR HAS BEEN DETECTED, NO NEED TO CHECK FURTHER)
00587                       *
00588A F78D 96 84     A           LDAA   PADDR    GET MSB OF ADDRESS
00589A F78F 84 F8     A           ANDA   #$F8     NOW REG. A HAS MSB OF STARTING ADDRESS OF CURRENT CHIP
00590A F791 8B 07     A           ADDA   #$07     CHANGE TO END OF CHIP
00591A F793 97 84     A           STAA   PADDR    SAVE IT
00592A F795 86 FF     A           LDAA   #$FF     END OF CHIP - MUST END WITH $FF
00593A F797 97 85     A           STAA   PADDR+1  SAVE IT
00594A F799 DE 84     A           LDX    PADDR    FOR FUTURE REFERENCE
00595A F79B 20 0E F7AB            BRA    R3
00596                       *
00597                       *        CHECK FOR THE END OF A 2K CHIP
```

```
PAGE  004  RAMCHK  .SA:1

00598                      *
00599A F79D 96 85     A R2      LDAA   PADDR+1  GET LSB OF ADDRESS
00600A F79F 81 FF     A         CMPA   #$FF     IS IT END OF A 256 BYTE PAGE?
00601A F7A1 26 29 F7CC          BNE    R6       NO - DEFINITELY NOT AND OF 2K
00602A F7A3 96 84     A         LDAA   PADDR    GET MSB OF ADDRESS
00603A F7A5 84 07     A         ANDA   #%00000111 GET RID OF BITS 3-7
00604A F7A7 81 07     A         CMPA   #%00000111 CHECK FOR 7 OR F
00605A F7A9 26 21 F7CC          BNE    R6       NOT END OF 2K
00606                      *
00607                      *      END OF 2K RAM CHIP - SHIFT RAM CHIP INDICATOR AND
00608                      *      TEST FOR END OF RAM
00609                      *
00610A F7AB 78 0082   A R3      ASL    PCHECK   SHIFT RAM BIT INDICATOR
00611A F7AE 79 0083   A         ROL    PCHECK+1
00612A F7B1 96 84     A         LDAA   PADDR    GET MSB OF ADDRESS
00613A F7B3 91 6E     A         CMPA   MRAM     END OF RAM?
00614A F7B5 26 15 F7CC          BNE    R6       NO
00615                      *
00616                      *      RAM TEST COMPLETE - INDICATE RESULTS IN ERCODE
00617                      *
00618A F7B7 96 6C     A         LDAA   MFLAG    RESET BIT - PREVENTS FURTHER RAM TESTS TODAY
00619A F7B9 84 BF     A         ANDA   #%10111111
00620A F7BB 97 6C     A         STAA   MFLAG    SAVE IT
00621A F7BD 96 44     A         LDAA   ERCODE   GET ERROR CODE BYTE
00622A F7BF DE 2E     A         LDX    RAMERR   ANY RAM ERRORS?
00623A F7C1 27 04 F7C7          BEQ    R4       NO -
00624A F7C3 8A 40     A         ORAA   #%01000000 YES - INDICATE RAM ERROR
00625A F7C5 20 02 F7C9          BRA    R5
00626A F7C7 84 BF     A R4      ANDA   #%10111111 INDICATE NO RAM ERRORS - RESET BIT
00627A F7C9 97 44     A R5      STAA   ERCODE   SAVE ERROR CODE
00628A F7CB 39                  RTS
00629                      *
00630                      *      ADDRESS NEXT BYTE - LOOP BACK TO BEGINNING IF THIS IS A POWER UP RAM TEST
00631                      *
00632A F7CC 08          R6      INX             ADDRESS NEXT RAM BYTE
00633A F7CD DF 84     A         STX    PADDR    SAVE ADDRESS
00634A F7CF 96 6C     A         LDAA   MFLAG    GET FLAGS
00635A F7D1 85 40     A         BITA   #%01000000 IS THIS A POWER UP RAM TEST?
00636A F7D3 27 8D F762          BEQ    R0       YES -
00637A F7D5 39                  RTS
00638                           END
TOTAL ERRORS 00000--00000
```

```
PAGE 001 EQUATE .SA:1

00498                         OPT    L
00499                  *
00500                  *         KEYBOARD ROUTINE
00501                  *
00502                  *
00503                  *
00504                  *             *****************************************************
00505                  *             *                                                   *
00506                  *             *   PROGRAMMER:   W.J. WIERCIENSKI                  *
00507                  *             *                                                   *
00508                  *             *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS   *
00509                  *             *   TRANSPORTATION TECHNOLOGY AND ENERGY            *
00510                  *             *   1201 WILSON AVENUE                              *
00511                  *             *   DOWNSVIEW, ONTARIO  M3M 1J8                     *
00512                  *             *                                                   *
00513                  *             *   TELEPHONE (416) 248-7141                        *
00514                  *             *                                                   *
00515                  *             *   NOVEMBER  1982 AD                               *
00516                  *             *                                                   *
00517                  *             *****************************************************
00518                  *
00519                  *
00520                  *
00521                  *         THIS ROUTINE HANDLES THE KEYBOARD DISPLAY AND KEYBOARD INPUT
00522                  *         PROCESSING IS CONTROLLED BY THE CURRENT STATE OF THE KEYBOARD,
00523                  *         AS INDICATED BY KYSTAT. THE FOLLOWING STATES ARE USED:
00524                  *
00525                  *         STATE 0: BLANK DISPLAY
00526                  *         STATE 1: TIME DISPLAY
00527                  *         STATE 2: WAITING FOR 2ND NIBBLE OF ADDRESS
00528                  *         STATE 3: WAITING FOR 3RD NIBBLE OF ADDRESS
00529                  *         STATE 4: WAITING FOR LAST NIBBLE OF ADDRESS
00530                  *         STATE 5: DATA DISPLAY (HEX)
00531                  *         STATE 6: WAITING FOR 1ST NIBBLE OF DATA
00532                  *         STATE 7: WAITING FOR 2ND NIBBLE OF DATA
00533                  *         STATE 8: DATA DISPLAY (HEX AND BINARY) - "F" MODE
00534                  *         STATE 9: POWER UP DISPLAY - SHOW RESULTS OF MEMORY TESTS
00535                  *
```

```
PAGE  002  KEYBD   .SA:1

00537A F800                     ORG    KEYBD
00538                   *
00539                   *         SEE IF ANY KEY IS BEING PRESSED
00540                   *
00541A F800 B6 5007  A          LDAA   PORT7    GET KEYBOARD PORT
00542A F803 84 1F    A          ANDA   #$1F     CLEAN IT UP
00543A F805 44                  LSRA            EXAMINE CONTROL BIT
00544A F806 25 05 F80D          BCS    L2       IT IS SET - A KEY HAS BEEN PRESSED
00545A F808 7F 0029  A          CLR    LSTKEY   IT IS NOT SET - CLEAR LAST KEY PRESS FLAG
00546A F80B 20 12 F81F          BRA    DISPLY   DO THE DISPLAY
00547                   *
00548                   *         PROCESS THE KEYPRESS
00549                   *
00550A F80D 7D 0029  A  L2      TST    LSTKEY   WAS KEY PRESSED 1/30 SEC AGO?
00551A F810 26 0D F81F          BNE    DISPLY   YES - IGNORE THIS KEY PRESS
00552A F812 7C 0029  A          INC    LSTKEY   SET LAST KEY PRESS FLAG TO 1
00553A F815 7F 002C  A          CLR    DISTMR   GIVE DISPLAY MORE LIFE
00554                   *
00555                   *         EXECUTE THE ROUTINE FOR A KEY PRESS WHILE IN CURRENT STATE
00556                   *         (KEY PUSHED IS NOW IN REG. A)
00557                   *
00558A F818 CE F856  A          LDX    #PSHTBL  START OF KEY PUSH TABLE
00559A F81B 8D 2D F84A          BSR    GETTBL   GET ENTRY FOR THIS STATE
00560A F81D AD 00    A          JSR    0,X      EXECUTE IT
00561                   *
00562                   *         DO DISPLAY - GET DISPLAY ROUTINE FOR THIS STATE FROM TABLE
00563                   *         AND EXECUTE IT
00564                   *
00565A F81F CE F86A  A DISPLY   LDX    #DSPTBL  START OF DISPLAY TABLE
00566A F822 8D 26 F84A          BSR    GETTBL   GET ENTRY FOR CURRENT STATE
00567A F824 AD 00    A          JSR    0,X      EXECUTE THE DISPLAY ROUTINE
00568                   *
00569                   *         MOVE THE DISPLAY BUFFERS TO OUTPUT PORTS
00570                   *
00571A F826 D6 28    A          LDAB   LITES2   GET LIGHTS 8 AND 9
00572A F828 0C                  CLC             PREPARE TO SHIFT THEM
00573A F829 56                  RORB            MOVE BITS 0 AND 1 TO 7 AND 6
00574A F82A 56                  RORB
00575A F82B 56                  RORB
00576A F82C D7 3C    A          STAB   TEMP06   SAVE LIGHT MASK
00577A F82E 86 20    A          LDAA   #$20     DEVICE SELECT - START AT BOTTOM LEFT DISPLAY
00578A F830 CE 0021  A          LDX    #DISP1   START AT DISP1
00579A F833 16          D1      TAB             GET SELECT BIT
00580A F834 DA 3C    A          ORAB   TEMP06   MERGE DEVICE SELECT AND LIGHTS (8,9) MASK
00581A F836 F7 5008  A          STAB   PORT8    SET DEVICE SELECT BITS
00582A F839 E6 00    A          LDAB   0,X      GET DATA FROM DISPLAY BUFFER
00583A F83B DA 2D    A          ORAB   DSPMSK   MERGE WITH DISPLAY MASK
00584A F83D F7 5009  A          STAB   PORT9    DISPLAY IT
00585A F840 08                  INX             NEXT WORD IN BUFFER
00586A F841 44                  LSRA            SHIFT DEVICE SELECT BIT - FOR NEXT DISPLAY
00587A F842 26 EF F833          BNE    D1
00588                   *       DISPLAY LIGHTS
00589A F844 96 27    A          LDAA   LITES    DISPLAY LIGHTS 0-7
00590A F846 B7 500A  A          STAA   PORTA    LIGHT THEM
00591A F849 39                  RTS
00592                   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00593                   *
00594                   *         FETCH THE 'KYSTAT'TH ENTRY FROM THE TABLE WHOSE
```

```
PAGE  003  KEYBD   .SA:1

00595                  *        BEGINNING ADDRESS IS IN REG. X
00596                  *         - ON EXIT, THE ENTRY IS IN REG. X AND REG. B IS 0
00597                  *.
00598                  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00599A F84A D6 20    A GETTBL LDAB   KYSTAT   GET STATE OF KEYBOARD
00600A F84C 27 05 F853        BEQ    FOUND    0 - ADDRESS ALREADY FOUND
00601A F84E 08         GET1   INX             INDEX TO NEXT TABLE ENTRY
00602A F84F 08                INX             (2 BYTES PER ENTRY)
00603A F850 5A                DECB            DECREMENT STATE
00604A F851 26 FB F84E        BNE    GET1     >0 - TRY AGAIN
00605A F853 EE 00    A FOUND  LDX    0,X      GET TABLE ENTRY
00606A F855 39                RTS
```

```
PAGE  004  KEYBD   .SA:1

00608                  *
00609                  *        KEY PUSH TABLE - ROUTINES TO BE EXECUTED WHEN A KEY HAS
00610                  *        BEEN PUSHED (ACTUAL KEY PUSHED IS PASSED IN REG. A)
00611                  *
00612A F856     F95B   A PSHTBL FDB     PUSH0    ROUTINE TO HANDLE KEY PUSH WHILE IN STATE 0
00613A F858     F963   A        FDB     PUSH1
00614A F85A     F974   A        FDB     PUSH2
00615A F85C     F974   A        FDB     PUSH3
00616A F85E     F97A   A        FDB     PUSH4
00617A F860     F99F   A        FDB     PUSH5
00618A F862     F9C9   A        FDB     PUSH6
00619A F864     F9CF   A        FDB     PUSH7
00620A F866     F9E1   A        FDB     PUSH8
00621A F868     F95B   A        FDB     PUSH9
00622                  *
00623                  *        DISPLAY TABLE - ROUTINES TO UPDATE THE APPROPRIATE DISPLAY
00624                  *        BUFFERS
00625                  *
00626A F86A     F87E   A DSPTBL FDB     DISPL0   DISPLAY WHILE IN STATE 0
00627A F86C     F888   A        FDB     DISPL1   DISPLAY WHILE IN STATE 1
00628A F86E     F8AC   A        FDB     DISPL2   DISPLAY WHILE IN STATE 2
00629A F870     F8B0   A        FDB     DISPL3   DISPLAY WHILE IN STATE 3
00630A F872     F8B4   A        FDB     DISPL4   DISPLAY WHILE IN STATE 4
00631A F874     F8B8   A        FDB     DISPL5   DISPLAY WHILE IN STATE 5
00632A F876     F8B0   A        FDB     DISPL6   DISPLAY WHILE ON STATE 6
00633A F878     F8B4   A        FDB     DISPL7   DISPLAY WHILE IN STATE 7
00634A F87A     F8B8   A        FDB     DISPL8   DISPLAY WHILE IN STATE 8
00635A F87C     F8D0   A        FDB     DISPL9   DISPLAY WHILE IN STATE 9
```

```
PAGE  005  KEYBD    .SA:1

00637                        *----------------------------------------
00638                        *       STATE 0 DISPLAY - BLANK OUT EVERYTHING
00639                        *----------------------------------------
00640A F87E 86 60      A DISPL0 LDAA   #$60      BLANK OUT UPPER AND LOWER DISPLAY
00641A F880 97 2D      A        STAA   DSPMSK    SAVE IN MASK
00642A F882 CE 0000    A        LDX    #0        BLANK OUT ALL LIGHTS
00643A F885 DF 27      A        STX    LITES
00644A F887 39                  RTS
00645                        *----------------------------------------
00646                        *       STATE 1 DISPLAY - TIME MODE
00647                        *----------------------------------------
00648A F888 96 09      A DISPL1 LDAA   HR        GET HOURS
00649A F88A BD F942    A        JSR    SPLIT     SPLIT IT
00650A F88D 97 26      A        STAA   DISP6     DISPLAY HOURS (MSN)
00651A F88F D7 25      A        STAB   DISP5     DISPLAY HOURS (LSN)
00652                        *
00653A F891 96 0A      A        LDAA   MIN       GET MINUTES
00654A F893 BD F942    A        JSR    SPLIT     SPLIT IT
00655A F896 97 21      A        STAA   DISP1     DISPLAY MINUTES (MSN)
00656A F898 D7 22      A        STAB   DISP2     DISPLAY MINUTES (LSN)
00657                        *
00658A F89A 96 0B      A        LDAA   SEC       GET SECONDS
00659A F89C BD F942    A        JSR    SPLIT     SPLIT IT
00660A F89F 97 23      A        STAA   DISP3     DISPLAY SECONDS (MSN)
00661A F8A1 D7 24      A        STAB   DISP4     DISPLAY SECONDS (LSN)
00662                        *
00663A F8A3 96 0E      A        LDAA   DAYPTR    GET DAY OF WEEK INDICATOR
00664A F8A5 48                  ASLA             SHOULD SET ONE OF LIGHTS 1-7
00665A F8A6 97 27      A        STAA   LITES     OUTPUT LIGHTS
00666A F8A8 7F 0028    A        CLR    LITES2    CLEAR LIGHTS 8 AND 9
00667A F8AB 39                  RTS
00668                        *----------------------------------------
00669                        *       STATE 2 DISPLAY - FLASH LIGHTS 1, 2 & 3
00670                        *----------------------------------------
00671A F8AC 86 0E      A DISPL2 LDAA   #%00001110 FOR LIGHTS 1,2 AND 3
00672A F8AE 20 73 F923          BRA    FLSHLT    FLASH THE LIGHTS
00673                        *----------------------------------------
00674                        *       STATE 3 AND STATE 6 DISPLAY - FLASH LIGHTS 1 & 2
00675                        *----------------------------------------
00676             F8B0 A DISPL6 EQU    *
00677A F8B0 86 06      A DISPL3 LDAA   #%00000110 FOR LIGHTS 1 & 2
00678A F8B2 20 6F F923          BRA    FLSHLT    FLASH THE LIGHTS
00679                        *----------------------------------------
00680                        *       STATE 4 AND STATE 7 DISPLAY - FLASH LIGHT 1
00681                        *----------------------------------------
00682             F8B4 A DISPL7 EQU    *
00683A F8B4 86 02      A DISPL4 LDAA   #%00000010 FOR LIGHT 1
00684A F8B6 20 6B F923          BRA    FLSHLT    FLASH THE LIGHT
00685                        *----------------------------------------
00686                        *       STATE 5 OR 8 DISPLAY - FETCH DATA FROM MEMORY AND
00687                        *       DISPLAY IT IN HEX - IF IN STATE 8 (F MODE), ALSO
00688                        *       DISPLAY IT IN BINARY ON THE LIGHTS
00689                        *----------------------------------------
00690             F8B8 A DISPL5 EQU    *
00691A F8B8 DE 2A      A DISPL8 LDX    ADDRSS    GET ADDRESS OF DATA
00692A F8BA A6 00      A        LDAA   0,X       GET DATA
00693A F8BC 97 3C      A        STAA   TEMP06    SAVE IT FOR DISPLAY LIGHTS
00694A F8BE BD F942    A        JSR    SPLIT     SPLIT DATA
```

```
PAGE  006  KEYBD    .SA:1

00695A F8C1 97 26      A        STAA   DISP6    DISPLAY DATA (MSN)
00696A F8C3 D7 25      A        STAB   DISP5    DISPLAY DATA (LSN)
00697A F8C5 96 20      A        LDAA   KYSTAT   GET STATE OF KEYBOARD
00698A F8C7 81 08      A        CMPA   #8       STATE = 8?
00699A F8C9 26 04 F8CF          BNE    D81      NO - END
00700                    *         WE ARE IN "F" MODE
00701A F8CB 96 3C      A        LDAA   TEMP06   GET DATA BACK
00702A F8CD 8D 5D F92C          BSR    ONLITE   PUT IT ON THE DISPLAY LIGHTS
00703A F8CF 39           D81    RTS
00704                    * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00705                    *         STATE 9 DISPLAY - SHOW RESULTS OF PROM CHECKSUM AND RAM TESTS
00706                    * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00707A F8D0 7C 002C    A DISPL9 INC    DISTMR   BUMP DISPLAY TIMER
00708A F8D3 26 04 F8D9          BNE    D91      STAY IN STATE 9
00709A F8D5 7F 0020    A        CLR    KYSTAT   GO TO STATE 0
00710A F8D8 39                  RTS
00711A F8D9 96 86      A D91    LDAA   PRERFL   ANY PROM ERRORS?
00712A F8DB 9A 2E      A        ORAA   RAMERR   OR RAM-ERRORS?
00713A F8DD 9A 2F      A        ORAA   RAMERR+1
00714A F8DF 27 2E F90F          BEQ    D92      NO -
00715                    *
00716                    *         PROM OR RAM ERROR - DISPLAY "BADD" AND LIGHT THE LIGHTS
00717                    *
00718A F8E1 CE 0B0A    A        LDX    #$0B0A   DISPLAY "BA" - FIRST PART OF "BADD"
00719A F8E4 DF 21      A        STX    DISP1
00720A F8E6 CE 0D0D    A        LDX    #$0D0D   DISPLAY "DD"
00721A F8E9 DF 23      A        STX    DISP3
00722A F8EB DE 2E      A        LDX    RAMERR   ANY RAM ERRORS?
00723A F8ED 27 08 F8F7          BEQ    D93      NO - DISPLAY PROM ERRORS
00724A F8EF 96 86      A        LDAA   PRERFL   ANY PROM ERRORS?
00725A F8F1 27 11 F904          BEQ    D94      NO - DISPLAY RAM ERRORS
00726                    * THERE ARE BOTH RAM AND PROM ERRORS !
00727A F8F3 D6 2C      A        LDAB   DISTMR   ONLY DISPLAY PROM ERRORS FOR 1/2 THE TIME
00728A F8F5 2B 0D F904          BMI    D94      DO RAM ERRORS IF TIMER > $7F
00729                    *
00730                    *         DISPLAY PROM ERRORS ON LIGHTS
00731                    *
00732A F8F7 CE 0000    A D93    LDX    #0       TURN OFF ALL LIGHTS
00733A F8FA DF 27      A        STX    LITES
00734A F8FC BD F92C    A        JSR    ONLITE   PUT PROM ERROR FLAG ON LIGHTS
00735A F8FF 86 20      A        LDAA   #$20     BLANK OUT UPPER DISPLAY
00736A F901 97 2D      A        STAA   DSPMSK
00737A F903 39                  RTS
00738                    *
00739                    *         DISPLAY RAM ERRORS ON LIGHTS
00740                    *
00741A F904 DF 27      A D94    STX    LITES    PUT RAM ERROR FLAGS ON LIGHTS (LIGHT 0 -> CHIP 1, ETC)
00742A F906 CE 0A0A    A        LDX    #$0A0A   PUT AA ON UPPER DISPLAY TO INDICATE RAM RESULTS
00743A F909 DF 25      A        STX    DISP5
00744A F90B 7F 002D    A        CLR    DSPMSK   ENABLE UPPER DISPLAY
00745A F90E 39                  RTS
00746                    *
00747                    *         NO PROM OR RAM ERRORS - DISPLAY "GOOD"
00748                    *
00749A F90F CE 0600    A D92    LDX    #$0600   DISPLAY "GO" - PART OF "GOOD"
00750A F912 DF 21      A        STX    DISP1
00751A F914 CE 000D    A        LDX    #$000D   DISPLAY "OD"
00752A F917 DF 23      A        STX    DISP3
```

```
PAGE  007  KEYBD    .SA:1

00753A F919 86 20     A         LDAA    #$20     BLANK OUT UPPER DISPLAY
00754A F91B 97 2D     A         STAA    DSPMSK
00755A F91D CE 0000   A         LDX     #0       BLANK OUT ALL LIGHTS
00756A F920 DF 27     A         STX     LITES
00757A F922 39                  RTS
```

```
PAGE  008  KEYBD   .SA:1

00759                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00760                         *
00761                         *         FLASH ANY OF THE LIGHTS 0 -> 7 AS INDICATED BY
00762                         *         THE MASK IN REGISTER A
00763                         *         - ON OUTPUT, REG. A AND REG. B ARE BOTH DESTROYED
00764                         *
00765                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00766A F923 D6 0C       A FLSHLT LDAB   TSEC       GET 1/10 SEC. COUNTER
00767A F925 54                   LSRB              EXAMINE BIT 0
00768A F926 25 01 F929           BCS    F1         CARRY SET - TURN ON LIGHTS
00769A F928 4F                   CLRA              CARRY NOT SET - TURN OFF ALL LIGHTS (0-7)
00770A F929 97 27       A F1     STAA   LITES      SAVE LIGHTS
00771A F92B 39                   RTS
00772                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00773                         *
00774                         *         DISPLAY THE CONTENTS OF REGISTER A ON THE DISPLAY
00775                         *          LIGHTS (BIT 0 -> LIGHT 1, ...)
00776                         *         - ON OUTPUT, BOTH REG. A AND REG. B ARE DESTROYED
00777                         *
00778                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00779A F92C D6 27       A ONLITE LDAB   LITES      GET DISPLAY LIGHTS 0-7
00780A F92E C4 01       A        ANDB   #$01       PRESERVE LIGHT 0
00781A F930 D7 27       A        STAB   LITES      SAVE IT
00782A F932 48                   ASLA              BITS 0-6 OF DATA TO LIGHTS 1-7
00783A F933 9A 27       A        ORAA   LITES      SET LITES 1-7 TO DATA
00784A F935 97 27       A        STAA   LITES      SAVE IT
00785A F937 C6 00       A        LDAB   #0         PREPARE TO SHIFT DATA BIT 7
00786A F939 59                   ROLB              REG. B NOW CONTAINS DATA BIT 7
00787A F93A 96 28       A        LDAA   LITES2     GET LIGHTS 8,9
00788A F93C 84 FE       A        ANDA   #$FE       PRESERVE LIGHT 9
00789A F93E 1B                   ABA               LIGHT 8 = DATA BIT 7
00790A F93F 97 28       A        STAA   LITES2     SAVE IT
00791A F941 39                   RTS
00792                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00793                         *
00794                         *         SPLIT REGISTER A INTO TWO NIBBLES
00795                         *         - ON OUTPUT, REG. A CONTAINS THE HIGH NIBBLE
00796                         *                      REG. B CONTAINS THE LOW NIBBLE
00797                         *
00798                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00799A F942 16                 SPLIT  TAB               SAVE DATA
00800A F943 44                        LSRA              REG. A GETS HIGH NIBBLE
00801A F944 44                        LSRA
00802A F945 44                        LSRA
00803A F946 44                        LSRA
00804A F947 C4 0F       A             ANDB   #$0F       REG. B GETS LOW NIBBLE
00805A F949 39                        RTS
00806                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00807                         *
00808                         *         MERGE TWO NIBBLES INTO 1 BYTE
00809                         *          - ON INPUT, REG. A CONTAINS THE HIGH NIBBLE
00810                         *                      REG. B CONTAINS THE LOW NIBBLE
00811                         *         - ON OUTPUT, REG. A CONTAINS THE RESULTING BYTE
00812                         *
00813                         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00814A F94A 48                 MERGE  ASLA              MOVE HIGH NIBBLE TO ITS PROPER LOCATION
00815A F94B 48                        ASLA
00816A F94C 48                        ASLA
```

```
PAGE  009  KEYBD    .SA:1

00817A F94D 48              ASLA
00818A F94E 1B              ABA          MERGE INTO 1 BYTE
00819A F94F 39              RTS
00820                  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00821                  *
00822                  *       SHIFT LOWER DISPLAY TO LEFT - ADD REG. A TO FAR RIGHT
00823                  *       - ON OUTPUT, REG. B IS DESTROYED
00824                  *                    REG. X IS DESTROYED
00825                  *
00826                  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00827A F950 DE 22    A SHFDSP LDX   DISP2    GET DISP2 AND DISP3
00828A F952 DF 21    A        STX   DISP1    SAVE THEM IN DISP1 AND DISP2
00829A F954 D6 24    A        LDAB  DISP4    GET DISP4
00830A F956 D7 23    A        STAB  DISP3    MOVE IT TO DISP3
00831A F958 97 24    A        STAA  DISP4    PUT REG. A IN DISP4
00832A F95A 39              RTS
```

```
PAGE  010  KEYBD    .SA:1

00834                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00835                       *         KEY PUSHED WHILE IN STATE 0 OR STATE 9
00836                       *         - GOTO TIME MODE
00837                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00838            F95B   A PUSH9   EQU    *
00839A F95B 86 01       A PUSH0   LDAA   #1        GO TO TIME STATE (1)
00840A F95D 97 20       A         STAA   KYSTAT
00841A F95F 7F 002D     A         CLR    DSPMSK    ENABLE UPPER AND LOWER DISPLAYS
00842A F962 39                    RTS
00843                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00844                       *         KEY PUSHED WHILE IN STATE 1
00845                       *         FIRST NIBBLE OF ADDRESS HAS JUST BEEN ENTERED
00846                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00847A F963 C6 20       A PUSH1   LDAB   #$20      DISABLE UPPER DISPLAY
00848A F965 D7 2D       A         STAB   DSPMSK    SAVE IT
00849A F967 CE 0000     A         LDX    #0        SET LOWER DISPLAY TO "0000"
00850A F96A DF 21       A         STX    DISP1
00851A F96C DF 23       A         STX    DISP3
00852A F96E 8D E0 F950            BSR    SHFDSP    SHIFT LOWER DISPLAY
00853A F970 7C 0020     A         INC    KYSTAT    GO TO STATE 2
00854A F973 39                    RTS
00855                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00856                       *         KEY PUSHED WHILE IN STATE 2 OR 3
00857                       *         - SAVE NEXT NIBBLE OF ADDRESS
00858                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00859            F974   A PUSH3   EQU    *
00860A F974 8D DA F950    PUSH2   BSR    SHFDSP    SHIFT LOWER DISPLAY
00861A F976 7C 0020     A         INC    KYSTAT    GO TO NEXT STATE (2->3 OR 3->4)
00862A F979 39                    RTS
00863                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00864                       *         KEY PUSHED WHILE IN STATE 4
00865                       *         - COMPLETE ADDRESS AND CHECK FOR READ PERMISSION
00866                       * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00867A F97A 8D D4 F950    PUSH4   BSR    SHFDSP    SHIFT LOWER DISPLAY
00868A F97C 7C 0020     A         INC    KYSTAT    GO TO STATE 5
00869                       *
00870                       *         ASSEMBLE THE LAST 4 KEY PUSHES INTO AN ADDRESS
00871                       *
00872A F97F 96 21       A         LDAA   DISP1     HIGH NIBBLE OF MSB OF ADDRESS
00873A F981 D6 22       A         LDAB   DISP2     LOW NIBBLE OF MSB OF ADDRESS
00874A F983 8D C5 F94A            BSR    MERGE     COMBINE THEM
00875A F985 97 2A       A         STAA   ADDRSS    SAVE MSB OF ADDRESS
00876A F987 96 23       A         LDAA   DISP3     HIGH NIBBLE OF LSB OF ADDRESS
00877A F989 D6 24       A         LDAB   DISP4     LOW NIBBLE OF LSB OF ADDRESS
00878A F98B 8D BD F94A            BSR    MERGE     COMBINE THEM
00879A F98D 97 2B       A         STAA   ADDRSS+1 SAVE LSB OF ADDRESS
00880                       *
00881A F98F 7F 0027     A         CLR    LITES     TURN OFF THE LIGHTS
00882A F992 7F 002D     A         CLR    DSPMSK    ENABLE UPPER AND LOWER DISPLAYS
00883                       *
00884                       *         ADDRSS NOW CONTAINS THE ADDRESS ENTERED AT THE KEYBOARD
00885                       *         - MAY THIS ADDRESS BE READ?
00886                       *
00887A F995 BD FA6D     A         JSR    READOK    IS READ PERMITTED?
00888A F998 24 04 F99E            BCC    P41       YES -
00889A F99A 86 01       A         LDAA   #1        NO- GO TO TIME MODE
00890A F99C 97 20       A         STAA   KYSTAT
00891A F99E 39              P41   RTS
```

```
PAGE  011  KEYBD   .SA:1

00892                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00893                          *          KEY PUSHED WHILE IN STATE 5
00894                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00895A F99F 81 09      A PUSH5  CMPA   #$09       KEY <= 9?
00896A F9A1 22 05 F9A8          BHI    P51        NO
00897A F9A3 86 01      A P5T01  LDAA   #1         GO TO TIME STATE (1)
00898A F9A5 97 20      A        STAA   KYSTAT
00899A F9A7 39                  RTS
00900A F9A8 81 0E      A P51    CMPA   #$0E       WANT TO ENTER DATA?
00901A F9AA 26 0E F9BA          BNE    P52        NO
00902                          *
00903                          *          THE "E" KEY WAS PUSHED - ENTER DATA
00904                          *
00905A F9AC BD FA87  A          JSR    WRITOK     IS CHANGE ALLOWED?
00906A F9AF 25 F2 F9A3          BCS    P5T01      NO - ENTER TIME STATE
00907A F9B1 CE 0000  A          LDX    #0         YES - SET UPPER DISPLAY TO "00"
00908A F9B4 DF 25    A          STX    DISP5
00909A F9B6 7C 0020  A          INC    KYSTAT     GO TO STATE 6
00910A F9B9 39                  RTS
00911A F9BA 23 09 F9C5 P52      BLS    P53        NOT "F" KEY
00912                          *
00913                          *          THE "F" KEY WAS PUSHED - GO TO "F" MODE
00914                          *
00915A F9BC 86 01      A        LDAA   #%00000001 TURN ON LIGHT 0 FOR "F" MODE
00916A F9BE 97 27      A        STAA   LITES      SAVE IT
00917A F9C0 86 08      A        LDAA   #8         GO TO STATE 8
00918A F9C2 97 20      A        STAA   KYSTAT
00919A F9C4 39                  RTS
00920                          *
00921                          *          ONE OF KEYS "A", "B", "C", AND "D" WAS PRESSED
00922                          *
00923A F9C5 BD FA30  A P53      JSR    ADDCHG     PROCESS ADDRESS CHANGE
00924A F9C8 39                  RTS
00925                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00926                          *          KEY PUSHED WHILE IN STATE 6
00927                          *          - FIRST NIBBLE OF DATA HAS BEEN ENTERED
00928                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00929A F9C9 97 25      A PUSH6  STAA   DISP5      DISPLAY NEW KEY PRESS
00930A F9CB 7C 0020  A          INC    KYSTAT     GO TO STATE 7
00931A F9CE 39                  RTS
00932                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00933                          *          KEY PUSHED WHILE IN STATE 7
00934                          *          - FULL DATA BYTE HAS BEEN ENTERED
00935                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00936A F9CF 16                 PUSH7  TAB               REG B NOW CONTAINS LOW NIBBLE OF ENTERED DATA
00937A F9D0 96 25      A        LDAA   DISP5      GET HIGH NIBBLE
00938A F9D2 BD F94A  A          JSR    MERGE      COMBINE THEM
00939A F9D5 DE 2A      A        LDX    ADDRSS     GET ADDRESS TO SAVE DATA
00940A F9D7 A7 00      A        STAA   0,X        SAVE DATA
00941A F9D9 7F 0027  A          CLR    LITES      TURN OFF LIGHTS 0->7
00942A F9DC 86 05      A        LDAA   #5         GO TO STATE 5
00943A F9DE 97 20      A        STAA   KYSTAT
00944A F9E0 39                  RTS
00945                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00946                          *          KEY PUSHED WHILE IN STATE 8 ("F" MODE)
00947                          * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00948A F9E1 81 08      A PUSH8  CMPA   #8         TRYING TO CHANGE MEMORY?
00949A F9E3 22 1D FA02          BHI    P84        NO
```

```
PAGE  012  KEYBD   .SA:1

00950                           *
00951                           *       ATTEMPT TO CHANGE MEMORY BIT  - IS IT ALLOWED?
00952                           *
00953A F9E5 BD FA87  A                JSR    WRITOK    WRITE ALLOWED?
00954A F9E8 24 05 F9EF                BCC    P81       YES - CHANGE ALLOWED
00955A F9EA 86 01     A L8TO1         LDAA   #1        GO TO TIME STATE
00956A F9EC 97 20     A               STAA   KYSTAT
00957A F9EE 39                        RTS
00958A F9EF DE 2A     A P81           LDX    ADDRSS    GET ADDRESS OF MEMORY TO CHANGE
00959A F9F1 4D                        TSTA             0 KEY PUSHED?
00960A F9F2 26 03 F9F7                BNE    P82       NO
00961                           *
00962                           *       "0" KEY PUSHED - ZERO OUT MEMORY
00963                           *
00964A F9F4 6F 00     A               CLR    0,X
00965A F9F6 39                        RTS
00966                           *
00967                           *       KEY 1-8 PUSHED - TOGGLE BIT OF MEMORY
00968                           *
00969A F9F7 5F          P82           CLRB             CONVERT REG. A TO A BIT POINTER
00970A F9F8 0D                        SEC              PREPARE TO ROTATE
00971A F9F9 59          P83           ROLB             SHIFT POINTER
00972A F9FA 4A                        DECA             DECREMENT KEY
00973A F9FB 26 FC F9F9                BNE    P83       DO AGAIN
00974A F9FD E8 00     A               EORB   0,X       TOGGLE THE REQUESTED BIT
00975A F9FF E7 00     A               STAB   0,X       SAVE THE RESULT
00976A FA01 39                        RTS
00977A FA02 81 09     A P84           CMPA   #9        "9" KEY PUSHED?
00978A FA04 27 E4 F9EA                BEQ    L8TO1     YES - GO TO TIME MODE
00979A FA06 81 0E     A               CMPA   #$0E      "E" KEY PUSHED?
00980A FA08 26 14 FA1E                BNE    P85       NO
00981                           *
00982                           *       "E" KEY PUSHED - ENTER DATA (IF ALLOWED)
00983                           *
00984A FA0A BD FA87   A               JSR    WRITOK    IS CHANGE ALLOWED?
00985A FA0D 25 DB F9EA                BCS    L8TO1     NO - GO TO TIME STATE
00986A FA0F CE 0000   A               LDX    #0
00987A FA12 DF 27     A               STX    LITES     BLANK OUT ALL LIGHTS
00988A FA14 CE 0000   A               LDX    #$0
00989A FA17 DF 25     A               STX    DISP5     SET UPPER DISPLAY TO "00"
00990A FA19 86 06     A               LDAA   #6        GO TO STATE 6
00991A FA1B 97 20     A               STAA   KYSTAT
00992A FA1D 39                        RTS
00993A FA1E 81 0F     A P85           CMPA   #$0F      F KEY PUSHED?
00994A FA20 26 0A FA2C                BNE    P86       NO
00995                           *
00996                           *       "F" KEY PUSHED - REVERT TO STATE 5
00997                           *
00998A FA22 CE 0000   A               LDX    #0
00999A FA25 DF 27     A               STX    LITES     BLANK OUT ALL LIGHTS
01000A FA27 86 05     A               LDAA   #5        GO TO STATE 5
01001A FA29 97 20     A               STAA   KYSTAT
01002A FA2B 39                        RTS
01003                           *
01004                           *       ONE OF KEYS "A", "B", "C", AND "D" WAS PRESSED
01005                           *
01006A FA2C BD FA30   A P86           JSR    ADDCHG    PROCESS ADDRESS CHANGE
01007A FA2F 39                        RTS
```

```
PAGE  013  KEYBD   .SA:1

01009                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
01010                    *
01011                    *        ONE OF THE KEYS "A", "B", "C" AND "D" HAS BEEN
01012                    *        PUSHED - ADJUST THE ADDRESS AND MAKE SURE THE NEW
01013                    *        ADDRESS MAY BE READ
01014                    *        - KEY "A": ADVANCE ADDRESS BY 1
01015                    *        - KEY "B": DECREMENT ADDRESS BY 1
01016                    *        - KEY "C": ADVANCE ADDRESS BY $10
01017                    *        - KEY "D": DECREMENT ADDRESS BY $10
01018                    *
01019                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
01020A FA30 CE FA65  A ADDCHG LDX    #ADDTBL  START OF ADD TABLE
01021A FA33 80 0A    A        SUBA   #$0A     REDUCE $A-$D TO $0-$3
01022A FA35 27 05 FA3C        BEQ    ADD2     ALREADY FOUND TABLE ENTRY
01023A FA37 08         ADD1   INX             ADDRESS NEXT TABLE ENTRY
01024A FA38 08                INX
01025A FA39 4A                DECA            DECREMENT COUNTER
01026A FA3A 26 FB FA37        BNE    ADD1     TRY AGAIN
01027                  *        X NOW POINTS TO AMOUNT BY WHICH ADDRESS IS INCREMENTED
01028A FA3C A6 01    A ADD2   LDAA   1,X      GET LSB OF ADJUSTMENT
01029A FA3E 9B 2B    A        ADDA   ADDRSS+1 ADD TO LSB OF ADDRESS
01030A FA40 97 2B    A        STAA   ADDRSS+1 SAVE NEW LSB OF ADDRESS
01031A FA42 A6 00    A        LDAA   0,X      GET MSB OF ADJUSTMENT
01032A FA44 99 2A    A        ADCA   ADDRSS   ADD MSB OF ADDRESS
01033A FA46 97 2A    A        STAA   ADDRSS   SAVE NEW MSB OF ADDRESS
01034                  *
01035                  *        MAKE SURE NEW ADDRESS HAS READ PERMISSION
01036                  *
01037A FA48 BD FA6D  A        JSR    READOK   NEW ADDRESSS OK TO READ?
01038A FA4B 24 05 FA52        BCC    ADD3     YES
01039A FA4D 86 01    A        LDAA   #1       NO - GO TO TIME MODE
01040A FA4F 97 20    A        STAA   KYSTAT
01041A FA51 39                RTS
01042                  *
01043                  *        PUT THE NEW ADDRESS ON DISPLAY
01044                  *
01045A FA52 96 2A    A ADD3   LDAA   ADDRSS   GET MSB OF ADDRESS
01046A FA54 BD F942  A        JSR    SPLIT    SPLIT IT UP
01047A FA57 97 21    A        STAA   DISP1    DISPLAY IT (MSN)
01048A FA59 D7 22    A        STAB   DISP2    (LSN)
01049A FA5B 96 2B    A        LDAA   ADDRSS+1 GET LSB OF ADDRESS
01050A FA5D BD F942  A        JSR    SPLIT    SPLIT IT UP
01051A FA60 97 23    A        STAA   DISP3    DISPLAY IT (MSN)
01052A FA62 D7 24    A        STAB   DISP4    (LSN)
01053A FA64 39                RTS
01054                  *
01055                  *        ADDRESS ADJUSTMENT TABLE - ENTRIES FOR KEYS "A" -> "D"
01056                  *
01057A FA65    0001  A ADDTBL FDB    $0001    "A" - ADD 1
01058A FA67    FFFF  A        FDB    $FFFF    "B" - SUBTRACT 1
01059A FA69    0010  A        FDB    $0010    "C" - ADD $10
01060A FA6B    FFF0  A        FDB    $FFF0    "D" - SUBTRACT $10
```

```
PAGE  014  KEYBD   .SA:1

01062                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
01063                    *.
01064                    *        DETERMINE IF IT IS PERMISSIBLE TO READ "ADDRSS"
01065                    *        - ON OUTPUT, CARRY BIT SET: READ IS PROHIBITED
01066                    *                     CARRY BIT CLEAR: READ IS ALLOWED
01067                    *                   REG. B IS DESTROYED
01068                    *
01069                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
01070                    *
01071                    *        DETERMINE IF LOCATION IS IN RAM
01072                    *
01073A FA6D D6 2A    A READOK LDAB   ADDRSS   GET MSB OF ADDRESS
01074A FA6F D1 6E    A        CMPB   MRAM     IS IT IN RAM?
01075A FA71 23 10 FA83        BLS    ROK      YES - READ IS OKAY
01076                    *
01077                    *        NOT IN RAM - IS IT IN PROM?
01078                    *
01079A FA73 C1 BF    A        CMPB   #$BF     PROM STARTS AT $C000
01080A FA75 22 0C FA83        BHI    ROK      IT IS PROM
01081                    *
01082                    *        NOT RAM, NOT PROM - IS IT A PORT?
01083                    *
01084A FA77 C1 50    A        CMPB   #$50     COULD IT BE A PORT?
01085A FA79 26 0A FA85        BNE    RNOTOK   NO
01086                    *      MIGHT BE A PORT, CHECK LSB OF ADDRESS
01087A FA7B D6 2B    A        LDAB   ADDRSS+1 GET LSB OF ADDRESS
01088A FA7D 27 06 FA85        BEQ    RNOTOK   ONLY PORTS $5001-$5006 ARE OK
01089A FA7F C1 06    A        CMPB   #$06     IS ADDRESS ABOVE $5006?
01090A FA81 22 02 FA85        BHI    RNOTOK   YES - READ NOT ALLOWED
01091                    *
01092                    *        READ IS PERMITTED
01093                    *
01094A FA83 0C            ROK    CLC             INDICATE READ IS ALLOWED
01095A FA84 39                   RTS
01096                    *
01097                    *        READ IS NOT ALLOWED
01098                    *
01099A FA85 0D            RNOTOK SEC             INDICATE READ IS PROHIBITED
01100A FA86 39                   RTS
```

```
PAGE  015  KEYBD   .SA:1

01102                     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
01103                     *
01104                     *         IS IT PERMITTED TO WRITE TO ADDRESS "ADDRSS"?
01105                     *         - ON OUTPUT, CARRY BIT SET: WRITE IS NOT ALLOWED
01106                     *                      CARRY BIT CLEAR: WRITE IS ALLOWED
01107                     *                      REG. B IS DESTROYED
01108                     *                      REG. X IS DESTROYED
01109                     *
01110                     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
01111                     *
01112                     *         MAKE SURE SECURITY CODE IS SET (UNLESS THE CODE
01113                     *         ITSELF IS BEING CHANGED)
01114                     *
01115A FA87 DE 2A    A WRITOK LDX    ADDRSS   GET ADDRESS
01116A FA89 8C 0001  A        CPX    #SECCOD  SECURITY CODE CHANGE?
01117A FA8C 27 2E FABC        BEQ    WOK      YES - ALWAYS ALLOWED
01118A FA8E 8C 0002  A        CPX    #SECCOD+1 SECURITY CODE CHANGE?
01119A FA91 27 29 FABC        BEQ    WOK      YES - ALWAYS ALLOWED
01120A FA93 DE 01    A        LDX    SECCOD   GET SECURITY CODE
01121A FA95 8C 0705  A        CPX    #$0705   IS IT SET?
01122A FA98 26 24 FABE        BNE    WNOTOK   NO - WRITE NOT ALLOWED
01123                     *
01124                     *         MAKE SURE ADDRESS IS WITHIN RAM
01125                     *
01126A FA9A D6 2A    A        LDAB   ADDRSS   GET MSB OF ADDRESS
01127A FA9C D1 6E    A        CMPB   MRAM     IN RAM?
01128A FA9E 22 1E FABE        BHI    WNOTOK   NO - WRITE IS NOT ALLOWED
01129                     *
01130                     *         ADDRESS IS IN RAM - PROTECT STACK
01131                     *
01132A FAA0 26 06 FAA8        BNE    W1       DEFINITELY NOT STACK
01133A FAA2 D6 2B    A        LDAB   ADDRSS+1 GET LSB OF ADDRESS
01134A FAA4 D1 6F    A        CMPB   MRAM+1   IS IT STACK?
01135A FAA6 22 16 FABE        BHI    WNOTOK   YES - WRITE PROHIBITED
01136                     *
01137                     *         IN RAM BUT NOT STACK - IS LOCATION PROTECTED?
01138                     *
01139A FAA8 CE FAC0  A W1     LDX    #PROTEC  START OF PROTECTED LOCATIONS TABLE
01140A FAAB DF 3C    A W2     STX    TEMP06   SAVE POINTER
01141A FAAD EE 00    A        LDX    0,X      GET PROTECTED ADDRESS
01142A FAAF 9C 2A    A        CPX    ADDRSS   SAME AS ADDRESS TO BE CHANGED?
01143A FAB1 27 0B FABE        BEQ    WNOTOK   YES - WRITE IS PROHIBITED
01144A FAB3 DE 3C    A        LDX    TEMP06   GET BACK POINTER
01145A FAB5 08                INX             POINT TO NEXT ENTRY
01146A FAB6 08                INX
01147A FAB7 8C FAD4  A        CPX    #EPROT   END OF TABLE?
01148A FABA 26 EF FAAB        BNE    W2       NO - TRY AGAIN
01149                     *
01150                     *         WRITE IS PERMITTED
01151                     *
01152A FABC 0C               WOK    CLC             INDICATE WRITE IS PERMITTED
01153A FABD 39                      RTS
01154                     *
01155                     *         WRITE IS PROHIBITED
01156                     *
01157A FABE 0D               WNOTOK SEC             INDICATE WRITE IS NOT ALLOWED
01158A FABF 39                      RTS
01159                     *
```

```
---

PAGE  016  KEYBD   .SA:1

01160                   *         TABLE OF PROTECTED MEMORY LOCATIONS
01161                 . *
01162A FAC0    0033 A PROTEC FDB    TMFLG1
01163A FAC2    0034 A        FDB    TMFLG2
01164A FAC4    006E A        FDB    MRAM
01165A FAC6    006F A        FDB    MRAM+1
01166A FAC8    0035 A        FDB    EXFLG
01167A FACA    0038 A        FDB    EXINDX
01168A FACC    0039 A        FDB    EXINDX+1
01169A FACE    003A A        FDB    SBRINX
01170A FAD0    003B A        FDB    SBRINX+1
01171A FAD2    0020 A        FDB    KYSTAT
01172          FAD4 A EPROT  EQU    *         END OF TABLE
01173                        END
TOTAL ERRORS 00000--00000
```

```
       OPT    L
       NAM    CLOCK
*
*         CLOCK ROUTINE
*
*
*             ***************************************************
*             *                                                 *
*             *   PROGRAMMER:   W.J. WIERCIENSKI                *
*             *                                                 *
*             *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS *
*             *   TRANSPORTATION TECHNOLOGY AND ENERGY          *
*             *   1201 WILSON AVENUE                            *
*             *   DOWNSVIEW, ONTARIO  M3M 1J8                   *
*             *                                                 *
*             *   TELEPHONE (416) 248-7141                      *
*             *                                                 *
*             *   NOVEMBER  1982 AD                             *
*             *                                                 *
*             ***************************************************
*
*
*        THIS ROUTINE CONTROLS THE CONTROLLER TIMING AND UPDATES THE TIME-OF-DAY
*        DATA.  IT IS ACTIVATED AT EACH REAL TIME CLOCK TICK BY THE
*        MASKABLE INTERRUPT.
*
```

```
PAGE  002  CLOCK   .SA:1  CLOCK

00525A FB00                       ORG    ITEST
00526                      *
00527                      *        DETERMINE TYPE OF INTERRUPT - COMM. OR CLOCK
00528                      *
00529A FB00 B6 6000  A            LDAA   ACIACS   GET ACIA STATUS REG.
00530A FB03 85 80    A            BITA   #$80     COMM. INTERRUPT??
00531A FB05 27 03 FB0A            BEQ    CLOCK    NO -- MUST BE THE CLOCK
00532A FB07 7E F000  A            JMP    COMM     GO TO COMMUNICATIONS ROUTINE
00533                      *
00534                      *        START OF CLOCK SUBROUTINE
00535                      *
00536                      *        NOTE: IN THIS ROUTINE TIME VARIABLES (HR, MIN, ETC.) ARE NOT SAVED
00537                      *        IMMEDIATELY AFTER BEING INCREMENTED - THIS IS IN CASE THE POWER GOES
00538                      *        DOWN BEFORE BOUNDARY CONDITIONS ARE CHECKED (IE, 60 SEC, 1 HOUR, ETC)
00539                      *        - THIS WAY "POWRON" WON'T RESET TIME VARIABLES IF POWER FAILS AT AN
00540                      *        INAPPROPRIATE MOMENT
00541                      *
00542A FB0A                       ORG    CLOCK
00543A FB0A 7F 5FFF  A            CLR    RTCLOK   RESET REAL TIME CLOCK
00544                      *
00545                      *        SET APPROPRIATE SHORT TIME FLAGS AND INCREMENT TIME COUNTERS
00546                      *
00547A FB0D D6 33    A            LDAB   TMFLG1   GET TIME FLAGS
00548A FB0F CA 01    A            ORAB   #TBIT0   SET 60HZ FLAG
00549A FB11 96 30    A            LDAA   PUCR     GET 60HZ PULSE COUNT
00550A FB13 4C                    INCA            INCREMENT IT
00551                      *
00552                      *        SEE IF 1/30TH SEC. HAS ELAPSED
00553                      *
00554A FB14 46                    RORA            LOOK AT LOW BIT - 1/30 SEC. YET?
00555A FB15 24 02 FB19            BCC    L1       NO -- GO ON
00556A FB17 CA 02    A            ORAB   #TBIT1   YES -- SET 30HZ FLAG
00557                      *
00558                      *        SEE IF 1/20TH SEC. HAS ELAPSED
00559                      *
00560A FB19 49             L1     ROLA            RESTORE PULSE COUNT
00561A FB1A 81 03    A            CMPA   #3       1/20 (IE, 3/60) SEC. YET?
00562A FB1C 26 02 FB20            BNE    L2       NO -- GO ON
00563A FB1E CA 04    A            ORAB   #TBIT2   YES -- SET 20HZ FLAG
00564                      *
00565                      *        SEE IF 1/10TH SEC. HAS ELAPSED
00566                      *
00567A FB20 81 05    A L2         CMPA   #5       1/10 (IE, 6/60) SECOND YET?
00568A FB22 22 04 FB28            BHI    L2A      YES -
00569A FB24 97 30    A            STAA   PUCR     SAVE 60HZ PULSE COUNT
00570A FB26 20 5C FB84            BRA    OUT1     LEAVE
00571A FB28 CA 0C    A L2A        ORAB   #TBIT2+TBIT3 YES -- SET 20HZ & 10HZ FLAGS
00572A FB2A 7F 0030  A            CLR    PUCR     CLEAR 60HZ PULSE COUNT
00573                      *
00574                      *        SEE IF 1/2 SEC. HAS ELAPSED
00575                      *
00576A FB2D 96 0C    A            LDAA   TSEC     GET 1/10SEC COUNT
00577A FB2F 4C                    INCA            INCREMENT IT
00578A FB30 81 05    A            CMPA   #5       1/2 (IE, 5/10) SEC. YET?
00579A FB32 26 06 FB3A            BNE    L3       NO -- GO ON
00580A FB34 CA 10    A            ORAB   #TBIT4   YES -- SET 1/2 SEC. FLAG
00581A FB36 97 0C    A            STAA   TSEC     SAVE 1/10SEC COUNTER
00582A FB38 20 4A FB84            BRA    OUT1     & LEAVE
```

```
PAGE  003  CLOCK   .SA:1  CLOCK

00583                         *
00584                         *       SEE IF 1 SECOND HAS ELAPSED
00585                         *.
00586A FB3A 81 09      A L3       CMPA   #9        1 SEC. YET?
00587A FB3C 22 04 FB42            BHI    L3A       YES -
00588A FB3E 97 0C      A          STAA   TSEC      NO - SAVE 1/10 SEC COUNTER
00589A FB40 20 42 FB84            BRA    OUT1      LEAVE
00590A FB42 CA 30      A L3A      ORAB   #TBIT4+TBIT5 SET 1/2 SEC. & 1 SEC. FLAGS
00591A FB44 7F 000C    A          CLR    TSEC      CLEAR 1/10 SEC. TIMER
00592A FB47 86 01      A          LDAA   #1
00593A FB49 9B 0B      A          ADDA   SEC       INCREMENT SECONDS
00594A FB4B 19                    DAA              CONVERT TO A BCD VALUE
00595A FB4C 97 3E      A          STAA   TEMP01    SAVE SECONDS (NOT IN SEC - IN CASE POWER FAILS)
00596                         *
00597                         *       SEE IF 6 SEC. HAVE ELAPSED
00598                         *
00599A FB4E 96 31      A          LDAA   SSEC      GET 6 SEC. COUNT
00600A FB50 4C                    INCA             INCREMENT IT
00601A FB51 81 05      A          CMPA   #5        6 SECONDS ELAPSED?
00602A FB53 22 04 FB59            BHI    L3B       YES -
00603A FB55 97 31      A          STAA   SSEC      NO - SAVE 6 SEC. COUNT
00604A FB57 20 13 FB6C            BRA    L4        CONTINUE
00605A FB59 CA 40      A L3B      ORAB   #TBIT6    SET 6 SECOND FLAG
00606A FB5B 7F 0031    A          CLR    SSEC      CLEAR 6 SECOND COUNT
00607                         *
00608                         *       SET DAY POINTER TO MATCH DAY OF WEEK.
00609                         *
00610A FB5E 96 08      A          LDAA   DAOFWK    GET DAY OF WEEK
00611A FB60 D7 33      A          STAB   TMFLG1    SAVE TIME FLAGS TO FREE B REG FOR CALCS
00612A FB62 5F                    CLRB
00613A FB63 0D                    SEC              MAKE MASK BIT .
00614A FB64 59           PLLO     ROLB             SHIFT BIT
00615A FB65 4A                    DECA             REDUCE DAY OF WEEK
00616A FB66 26 FC FB64            BNE    PLLO      NOT DONE YET
00617A FB68 D7 0E      A          STAB   DAYPTR    SAVE DAY POINTER
00618A FB6A D6 33      A          LDAB   TMFLG1    GET FLAGS BACK
00619                         *
00620                         *       SEE IF 30 SEC. HAVE ELAPSED
00621                         *
00622A FB6C 96 3E      A L4       LDAA   TEMP01    GET SECONDS
00623A FB6E 81 30      A          CMPA   #$30      30 SEC. ELAPSED?
00624A FB70 26 15 FB87            BNE    L5        NO - GO ON
00625A FB72 CA 80      A          ORAB   #TBIT7    YES -- SET 30 SECOND FLAG
00626                         *
00627A FB74 7C 002C    A          INC    DISTMR    BUMP DISPLAY TIMER
00628A FB77 26 09 FB82            BNE    OUT3      DISPLAY TIMER NOT TIMED OUT YET
00629                         *
00630                         *   DISPLAY TIMER TIMED OUT (NO KEY HIT FOR 255 MINUTES)
00631                         *
00632A FB79 7F 0001    A          CLR    SECCOD    CLEAR SECURITY CODE
00633A FB7C 7F 0002    A          CLR    SECCOD+1  CLEAR SECURITY CODE
00634A FB7F 7F 0020    A          CLR    KYSTAT    GO TO KEYBOARD STATE 0 (BLANK OUT DISPLAY)
00635                         *
00636A FB82 97 0B      A OUT3     STAA   SEC       SAVE SECONDS
00637A FB84 D7 33      A OUT1     STAB   TMFLG1    STORE SHORT TIME FLAGS
00638A FB86 3B                    RTI
00639                         *
00640                         *       SEE IF 1 MIN. HAS ELAPSED
```

```
PAGE  004  CLOCK   .SA:1  CLOCK

00641                     *
00642A FB87 81 59     A L5      CMPA   #$59      HAS 1 MIN. GONE?
00643A FB89 23 F7 FB82  .       BLS    OUT3      NO -- LEAVE
00644                     *
00645                     *    A FULL MINUTE HAS BEEN REACHED
00646                     *
00647A FB8B CA 80     A         ORAB   #TBIT7    YES -- SET 30 SEC. FLAG
00648A FB8D D7 33     A         STAB   TMFLG1    & SAVE SHORT TIME FLAGS
00649A FB8F D6 34     A         LDAB   TMFLG2    GET LONG TIME FLAG WORD
00650A FB91 CA 01     A         ORAB   #TBIT0    SET 1 MIN FLAG
00651A FB93 7F 000B   A         CLR    SEC       CLEAR SECONDS
```

```
PAGE  005  CLOCK   .SA:1  CLOCK

00653                         *-----------------------------------------------------------
00654                         *
00655                         *        SWI  -  SOFTWARE INTERRUPT
00656                         *
00657                         *-----------------------------------------------------------
00658                         *
00659A FB96                          ORG    SWI
00660A FB96 7F 0031   A              CLR    SSEC      CLEAR 6 SECOND COUNT
00661A FB99 86 01     A              LDAA   #1
00662A FB9B 9B 0A     A              ADDA   MIN       INCREMENT MINUTES
00663A FB9D 19                       DAA              CONVERT TO BCD
00664A FB9E 97 3E     A              STAA   TEMP01    STORE MINUTES (NOT IN MIN - IN CASE POWER FAILS)
00665                         *
00666                         *       SEE IF 20 MINUTES HAVE ELAPSED
00667                         *
00668A FBA0 81 20     A              CMPA   #$20      20 MINUTES YET?
00669A FBA2 27 04 FBA8               BEQ    L6        YES --
00670A FBA4 81 40     A              CMPA   #$40      HOW ABOUT NOW?
00671A FBA6 26 02 FBAA               BNE    L6A       NO --
00672A FBA8 CA 08     A L6           ORAB   #TBIT3    SET 20 MIN. FLAG
00673                         *
00674                         *       SEE IF 15 MINUTES HAVE ELAPSED
00675                         *
00676A FBAA 81 15     A L6A          CMPA   #$15      15 MINUTES?
00677A FBAC 27 04 FBB2               BEQ    L6B       YES --
00678A FBAE 81 45     A              CMPA   #$45      45 MINUTES?
00679A FBB0 26 02 FBB4               BNE    L6C       NO --
00680A FBB2 CA 04     A L6B          ORAB   #TBIT2    SET 15 MIN FLAG
00681                         *
00682                         *       SEE IF 30 MINUTES HAVE ELAPSED
00683                         *
00684A FBB4 81 30     A L6C          CMPA   #$30      30 MINUTES YET?
00685A FBB6 26 02 FBBA               BNE    L7        NO --
00686A FBB8 CA 14     A              ORAB   #TBIT2+TBIT4 YES -- SET 15,30 MIN. FLAGS
00687                         *
00688                         *       SEE IF 6 MINUTES HAVE ELAPSED
00689                         *
00690A FBBA 96 32     A L7           LDAA   SMIN      GET 6 MINUTE COUNT
00691A FBBC 4C                       INCA             INCREMENT IT
00692A FBBD 81 05     A              CMPA   #5        6 MIN. ELAPSED?
00693A FBBF 22 04 FBC5               BHI    L7B       YES -
00694A FBC1 97 32     A              STAA   SMIN      NO - SAVE 6 MIN. COUNT
00695A FBC3 20 05 FBCA               BRA    L7A       AND CONTINUE
00696A FBC5 CA 02     A L7B          ORAB   #TBIT1    YES -- SET FLAG
00697A FBC7 7F 0032   A              CLR    SMIN      CLEAR COUNT
00698                         *
00699                         *       SEE IF 1 HOUR HAS ELAPSED
00700                         *
00701A FBCA 96 3E     A L7A          LDAA   TEMP01    GET MINUTES
00702A FBCC 81 59     A              CMPA   #$59      HAS AN HOUR GONE BY?
00703A FBCE 22 04 FBD4               BHI    L7C       YES -
00704A FBD0 97 0A     A              STAA   MIN       NO - SAVE MINUTES
00705A FBD2 20 69 FC3D               BRA    OUT2      AND LEAVE
00706A FBD4 CA 3E     A L7C          ORAB   #TBIT1+TBIT2+TBIT3+TBIT4+TBIT5 YES -- SET 1 HOUR FLAGS
00707A FBD6 7F 000A   A              CLR    MIN       CLEAR THE MINUTES
00708A FBD9 7F 0032   A              CLR    SMIN      RESET 6 MIN. COUNT
00709A FBDC 86 01     A              LDAA   #1
00710A FBDE 9B 09     A              ADDA   HR        BUMP THE HOURS
```

```
PAGE  006  CLOCK   .SA:1  CLOCK

00711A FBE0 19                DAA             CONVERT TO BCD
00712A FBE1 97 3E    A        STAA   TEMP01   SAVE HOURS (NOT IN HR - IN CASE POWER FAILS)
00713                    *
00714                    *       CHECK FOR DAYLIGHT SAVINGS TIME ADJUSTMENT
00715                    *
00716A FBE3 81 02    A        CMPA   #02      IS IT 0200 HOURS?
00717A FBE5 26 13 FBFA        BNE    L10      NO --
00718A FBE7 96 08    A        LDAA   DAOFWK   YES -
00719A FBE9 81 07    A        CMPA   #07      IS IT SUNDAY?
00720A FBEB 26 0B FBF8        BNE    L9       NO --
00721A FBED 7D 000F  A        TST    TIMEFX   YES -- HAS TIME BEEN FIXED?
00722A FBF0 27 06 FBF8        BEQ    L9       YES -- GO ON
00723A FBF2 BD FCA0  A        JSR    DAYLIT   NO -- CHECK FOR DAYLIGHT SAVINGS ADJUSTMENT
00724A FBF5 7F 000F  A        CLR    TIMEFX   RESET TIME CHANGE FLAG
00725A FBF8 96 3E    A L9     LDAA   TEMP01   GET HOURS
00726                    *
00727                    *       CHECK FOR THE END OF THE DAY
00728                    *
00729A FBFA 81 23    A L10    CMPA   #$23     END-OF-DAY?
00730A FBFC 22 04 FC02        BHI    L10A     YES - CONTINUE
00731A FBFE 97 09    A        STAA   HR       NO - SAVE HOURS
00732A FC00 20 3B FC3D        BRA    OUT2     AND LEAVE
00733A FC02 CA C0    A L10A   ORAB   #TBIT6+TBIT7 YES -- SET END-OF-DAY FLAGS
00734A FC04 D7 0F    A        STAB   TIMEFX   ENABLE DAYLIGHT SAVINGS ADJUSTMENT (NO MANUAL INITIALIZATION NEEDED)
00735A FC06 7F 0009  A        CLR    HR       CLEAR HOURS
00736A FC09 7C 0008  A        INC    DAOFWK   INCREMENT DAY OF WEEK
00737A FC0C 78 000E  A        ASL    DAYPTR   SHIFT DAY OF WEEK POINTER
00738A FC0F 96 05    A        LDAA   DAOFMO   GET DAY OF MONTH
00739A FC11 8B 01    A        ADDA   #1       BUMP IT
00740A FC13 19                DAA             CONVERT TO A BCD VALUE
00741A FC14 97 05    A        STAA   DAOFMO   SAVE IT
00742A FC16 96 07    A        LDAA   DOYL     GET LSB DAY OF YEAR
00743A FC18 8B 01    A        ADDA   #1       BUMP IT
00744A FC1A 19                DAA             CONVERT TO BCD
00745A FC1B 97 07    A        STAA   DOYL     SAVE IT
00746A FC1D 26 03 FC22        BNE    L30      NOT ZERO -- DO NOT CHANGE MSB
00747A FC1F 7C 0006  A        INC    DOYM     DAY OF YEAR (MSB)
00748                    *
00749                    *       CHECK FOR THE END OF THE WEEK
00750                    *
00751A FC22 86 08    A L30    LDAA   #08
00752A FC24 91 08    A        CMPA   DAOFWK   IS IT END OF WEEK?
00753A FC26 26 06 FC2E        BNE    MON      NO -- CHECK FOR END OF MONTH
00754A FC28 86 01    A        LDAA   #01      YES --
00755A FC2A 97 08    A        STAA   DAOFWK   RESET DAY OF WEEK
00756A FC2C 97 0E    A        STAA   DAYPTR   & DAY POINTER
00757                    *
00758                    *       CHECK FOR THE END OF THE MONTH
00759                    *
00760A FC2E D7 34    A MON    STAB   TMFLG2   SAVE LONG TIME FLAGS TO FREE B FOR CALCS
00761A FC30 D6 04    A        LDAB   MONTH    GET MONTH
00762                    *
00763A FC32 8D 2F FC63        BSR    NOFDAY   THIS S/R RETURNS # OF DAYS IN MONTH IN ACC A
00764                    *              WHILE ACC B HAS THE MONTH #  (ALSO SETS DIM)
00765                    *
00766A FC34 8B 01    A        ADDA   #1       BUMP DAY NUMBER
00767A FC36 19                DAA             CONVERT TO BCD
00768A FC37 91 05    A        CMPA   DAOFMO   MONTH END YET??
```

```
PAGE  007  CLOCK    .SA:1  CLOCK

00769A FC39 27 05 FC40         BEQ    ENDMON   YES --
00770A FC3B D6 34    A         LDAB   TMFLG2   GET BACK TIME FLAGS
00771A FC3D D7 34    A OUT2    STAB   TMFLG2   SAVE LONG TIME FLAGS
00772A FC3F 3B         OUT     RTI
00773                  *
00774                  *      END OF MONTH FOUND (MONTH # IS IN REG B)
00775A FC40 86 01    A ENDMON  LDAA   #1
00776A FC42 97 05    A         STAA   DAOFMO   RESET DAY OF MONTH
00777A FC44 17                 TBA             MONTH TO A
00778A FC45 8B 01    A         ADDA   #1       BUMP MONTH
00779A FC47 19                 DAA             CONVERT TO BCD
00780A FC48 16                 TAB             MONTH BACK TO B
00781A FC49 86 01    A         LDAA   #1
00782                  *
00783                  *       CHECK FOR END OF YEAR
00784                  *
00785A FC4B C1 13    A         CMPB   #$13     END OF YEAR??
00786A FC4D 27 06 FC55         BEQ    ENDYR    YES --
00787A FC4F 8D 12 FC63 NO2     BSR    NOFDAY   NO -- GET NO. OF DAYS THIS MONTH
00788A FC51 D7 04    A         STAB   MONTH    SAVE MONTH NO.
00789A FC53 20 EA FC3F         BRA    OUT      & LEAVE
00790                  *      END OF YEAR FOUND
00791A FC55 97 07    A ENDYR   STAA   DOYL     SET DAY OF YEAR TO 1
00792A FC57 7F 0006  A         CLR    DOYM
00793A FC5A 9B 03    A         ADDA   YEAR     ADD TO YEAR
00794A FC5C 19                 DAA             CONVERT TO BCD
00795A FC5D 97 03    A         STAA   YEAR     SAVE YEAR
00796A FC5F C6 01    A         LDAB   #1       1 TO B FOR MONTH NO
00797A FC61 20 EC FC4F         BRA    NO2
```

```
PAGE  008  CLOCK    .SA:1  CLOCK

00799                      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00800                      *
00801                      *  ON ENTRY TO S/R NOFDAY , ACC B CONTAINS MONTH #, ACC A IS UNPREDICTABLE
00802                      *  ON EXIT ACC B IS UNCHANGED WHILE ACC A CONTAINS # OF DAYS IN MONTH
00803                      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00804A FC63 CE FC8E   A NOFDAY LDX    #MONTBL
00805A FC66 D7 3E     A        STAB   TEMP01    SAVE MONTH NO.
00806                      *
00807                      *        GET CORRECT VALUE FROM MONTH TABLE
00808                      *
00809A FC68 A6 00     A MONLP  LDAA   0,X       GET DAYS IN MONTH
00810A FC6A 5A                 DECB             DECREMENT MONTH NUMBER
00811A FC6B 27 03 FC70         BEQ    GOTIT     CORRECT TABLE ENTRY -- GO ON
00812A FC6D 08                 INX              NOT RIGHT MONTH YET -- TRY NEXT ENTRY
00813A FC6E 20 F8 FC68         BRA    MONLP
00814                      *
00815A FC70 D6 3E     A GOTIT  LDAB   TEMP01    GET MONTH NO.
00816                      *
00817                      *        CHECK FOR LEAP YEAR
00818                      *
00819A FC72 C1 02     A        CMPB   #2        FEBRUARY?
00820A FC74 26 13 FC89         BNE    RTN       NO -- NO ADJUSTMENT NEEDED
00821                      *
00822A FC76 5F                 CLRB
00823A FC77 97 0D     A        STAA   DIM       SAVE DAYS IN MONTH (CORRECTED LATER IF LEAP YEAR)
00824A FC79 4F                 CLRA
00825A FC7A 8B 01     A YR1    ADDA   #1        ADD 1 TO YEAR COUNT
00826A FC7C 19                 DAA              CONVERT TO DECIMAL
00827A FC7D 5C                 INCB             BUMP BINARY,YEAR COUNT
00828A FC7E 91 03     A        CMPA   YEAR      DEC. YEAR COUNT MATCH EXISTING YEAR?
00829A FC80 26 F8 FC7A         BNE    YR1       NO -- ADD ANOTHER YEAR
00830A FC82 96 0D     A        LDAA   DIM       RESTORE DAYS IN MONTH
00831                      *
00832                      *  HERE ACC B HAS YEAR COUNT IN BINARY
00833                      *
00834A FC84 C5 03     A        BITB   #3        LEAP YEAR?
00835A FC86 26 01 FC89         BNE    RTN       NO --
00836A FC88 4C                 INCA             YES -- BUMP DAYS IN MONTH
00837A FC89 97 0D     A RTN    STAA   DIM       SAVE DAYS IN MONTH
00838A FC8B D6 3E     A        LDAB   TEMP01    GET BACK MONTH NUMBER
00839A FC8D 39                 RTS
00840                      *
00841                      *        MONTH TABLE - HAS NUMBER (BCD) OF DAYS IN EACH MONTH
00842                      *
00843A FC8E     31    A MONTBL FCB    $31       JAN
00844A FC8F     28    A        FCB    $28       FEB
00845A FC90     31    A        FCB    $31       MAR
00846A FC91     30    A        FCB    $30       APR
00847A FC92     31    A        FCB    $31       MAY
00848A FC93     30    A        FCB    $30       JUN
00849A FC94     31    A        FCB    $31       JUL
00850A FC95     31    A        FCB    $31       AUG
00851A FC96     30    A        FCB    $30       SEP
00852                      *
00853                      * FOLLOWING INSTRUCTIONS OCCUPY SIX (6) EMPTY SLOTS IN THE
00854                      * TABLE OF DAYS IN EACH MONTH. MUST OCCUPY 6 BYTES.
00855A FC97 01                 NOP
00856A FC98 01                 NOP
```

```
PAGE  009  CLOCK    .SA:1  CLOCK

00857A FC99 01                 NOP
00858A FC9A 01                 NOP
00859A FC9B 01                 NOP
00860A FC9C 01                 NOP
00861                     *
00862                     *
00863A FC9D    31   A          FCB    $31      OCT
00864A FC9E    30   A          FCB    $30      NOV
00865A FC9F    31   A          FCB    $31      DEC
```

```
PAGE  010  CLOCK   .SA:1  CLOCK

00867                      * * * * * * * * * * * * * * * * * * * * * * * *
00868                      *        MAKE DAYLIGHT SAVINGS ADJUSTMENT
00869                      * * * * * * * * * * * * * * * * * * * * * * * *
00870A FCA0 96 05    A DAYLIT LDAA   DAOFMO   GET DAY OF MONTH
00871A FCA2 8B 07    A        ADDA   #7       ADD 7
00872A FCA4 19                DAA             CONVERT TO BCD
00873A FCA5 91 0D    A        CMPA   DIM      LAST SUNDAY OF MONTH?
00874A FCA7 22 01 FCAA        BHI    SEASON   YES --
00875A FCA9 39                RTS             NO -- NO ADJUSTMENT NECESSARY
00876                      *
00877                      *  TO GET TO S/R SEASON IT MUST BE THE LAST SUNDAY IN A MONTH AND 2 AM
00878                      *
00879A FCAA 96 04    A SEASON LDAA   MONTH
00880A FCAC 81 04    A        CMPA   #04      APRIL?
00881A FCAE 27 05 FCB5        BEQ    SPRING   YES -- DO DAYLIGHT SAVINGS ADJUSTMENT
00882A FCB0 81 10    A        CMPA   #$10     OCTOBER?
00883A FCB2 27 05 FCB9        BEQ    FALL     YES -- DO DAYLIGHT SAVINGS ADJUSTMENT
00884A FCB4 39                RTS             NO -- NO ADJUSTMENT NECESSARY
00885                      *
00886                      * TIME CHANGE
00887                      *
00888A FCB5 7C 003E  A SPRING INC    TEMP01   ADVANCE THE CLOCK
00889A FCB8 39                RTS
00890                      *
00891A FCB9 7A 003E  A FALL   DEC    TEMP01   SET CLOCK BACK
00892A FCBC 39                RTS
00893                         END
TOTAL ERRORS 00000--00000
```

```
PAGE  001  EQUATE  .SA:1  POWRON

       OPT    L
       NAM    POWRON
*
*
*         POWERON SERVICE ROUTINE
*
*
*             ****************************************************
*             *                                                  *
*             *   PROGRAMMER:   W.J. WIERCIENSKI                 *
*             *                                                  *
*             *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS  *
*             *   TRANSPORTATION TECHNOLOCY AND ENERGY           *
*             *   1201 WILSON AVENUE                             *
*             *   DOWNSVIEW, ONTARIO  M3M 1J8                    *
*             *                                                  *
*             *   TELEPHONE (416) 248-7141                       *
*             *                                                  *
*             *   NOVEMBER  1982 AD                              *
*             *                                                  *
*             ****************************************************
*
*
*         THIS PROGRAM INITIALIZES CONTROLLER STATUS
*         WHEN POWER IS APPLIED
*
```

```
PAGE  002  POWRON  .SA:1  POWRON

00525A FCD0                     ORG    POWRON
00526                     *
00527                     *          DETERMINE END OF CONSECUTIVE RAM AND LOAD STACK POINTER
00528                     *
00529A FCD0 8E 03FF  A          LDS    #STAK1   STACK POINTER FOR 1K RAM
00530A FCD3 CE 03DF  A          LDX    #STAK1-$20 END OF USEFUL RAM
00531A FCD6 DF 6E    A          STX    MRAM     SAVE IT
00532A FCD8 CE 07FF  A          LDX    #STAK2   POINTER TO MAX ADDRESS OF 2K RAM
00533A FCDB 86 96    A          LDAA   #%10010110 PUT A RANDOM TEST VALUE INTO REGISTER A
00534A FCDD E6 00    A L01      LDAB   0,X      SAVE VALUE NOW PRESENT IN MEMORY
00535A FCDF A7 00    A          STAA   0,X      WRITE TO SUSPECTED END OF RAM
00536A FCE1 A6 00    A          LDAA   0,X      READ BACK
00537A FCE3 E7 00    A          STAB   0,X      RESTORE MEMORY
00538A FCE5 81 96    A          CMPA   #%10010110 COMPARE WITH VALUE SAVED THERE
00539A FCE7 26 1F FD08          BNE    L02      STACK POINTER AND END OF USEFUL RAM ARE SET
00540A FCE9 DF 3E    A          STX    TEMP01   SAVE TO TEMPORY LOCATION
00541A FCEB 9E 3E    A          LDS    TEMP01   LOAD NEW END OF RAM INTO STACK POINTER
00542A FCED C6 DF    A          LDAB   #$DF     SUBTRACT 32 BYTES ( $FF-$20=$DF )
00543A FCEF D7 3F    A          STAB   TEMP02   SAVE IT
00544A FCF1 DE 3E    A          LDX    TEMP01   END OF USEFUL RAM
00545A FCF3 DF 6E    A          STX    MRAM     SAVE IT
00546A FCF5 8C 4FDF  A          CPX    #EOFRAM  COMPARE WITH MAX END OF USEFUL RAM (20K-32 BYTES FOR STACK)
00547A FCF8 27 0E FD08          BEQ    L02      STACK POINTER AND END OF USEFUL RAM ARE SET
00548A FCFA C6 FF    A          LDAB   #$FF     END OF CHIP ADDRESS ALWAYS $FF
00549A FCFC D7 3F    A          STAB   TEMP02   SAVE IT
00550A FCFE D6 3E    A          LDAB   TEMP01   GET MSB OF ADDRESS
00551A FD00 CB 08    A          ADDB   #TWOK    ADD 2K MORE ( 2K=$0800 )
00552A FD02 D7 3E    A          STAB   TEMP01   SAVE NEW MSB OF STACK POINTER
00553A FD04 DE 3E    A          LDX    TEMP01   PUT NEW STACK POINTER INTO INDEX REGISTER
00554A FD06 20 D5 FCDD          BRA    L01      GO SEE IF MORE RAM IS AVAILABLE
```

```
PAGE  003  POWRON  .SA:1  POWRON

00556                  *
00557                  *
00558                  *        INITIALIZE COMMUNICATIONS - 300 BAUD,8 DATA BITS,EVEN PARITY,1 STOP BIT
00559                  *
00560                  *
00561A FD08 86 03    A L02    LDAA   #%00000011 MASTER RESET ACIA
00562A FD0A B7 6000  A        STAA   ACIACS
00563A FD0D 86 9A    A        LDAA   #%10011010 XMITTER OFF, RECEIVER ON
00564A FD0F B7 6000  A        STAA   ACIACS
00565A FD12 86 01    A        LDAA   #1         LOAD 1 TO ACCUMULATOR
00566A FD14 97 45    A        STAA   RBITS      RBITS=1 - BEGINING OF MSG EXPECTED
00567A FD16 7F 0046  A        CLR    XBITS      XBITS=0
```

```
PAGE  004  POWRON  .SA:1  POWRON

00569                      *
00570                      *         CHECK DATE AND TIME VARIABLES - MAKE SURE THEY ARE VALID (IF NOT, RESET THEM)
00571                      *.
00572                      *         CHECK 1/60TH SECOND COUNTER
00573A FD19 96 30    A          LDAA   PUCR     GET 1/60TH SEC. COUNTER
00574A FD1B 81 05    A          CMPA   #5       MAX. VALUE IS 5
00575A FD1D 22 33 FD52          BHI    L03      TOO LARGE - CLEAR TIME VALUES
00576                      *         CHECK 1/10TH SECOND COUNTER
00577A FD1F 96 0C    A          LDAA   TSEC     GET 1/10TH SEC. COUNTER
00578A FD21 81 09    A          CMPA   #9       MAX. VALUE IS 9
00579A FD23 22 2D FD52          BHI    L03      TOO LARGE - CLEAR TIME VALUES
00580                      *         CHECK 6 SECOND COUNT
00581A FD25 96 31    A          LDAA   SSEC     GET 6 SECOND COUNTER
00582A FD27 81 05    A          CMPA   #5       MAX. VALUE IS 5
00583A FD29 22 27 FD52          BHI    L03      TOO LARGE - CLEAR TIME VALUES
00584                      *         CHECK 6 MINUTE COUNT
00585A FD2B 96 32    A          LDAA   SMIN     GET 6 MINUTE COUNT
00586A FD2D 81 05    A          CMPA   #5       MAX. VALUE IS 5
00587A FD2F 22 21 FD52          BHI    L03      TOO LARGE - CLEAR TIME VALUES
00588                      *         CHECK SECONDS COUNT
00589A FD31 96 0B    A          LDAA   SEC      GET SECONDS COUNT
00590A FD33 81 59    A          CMPA   #$59     MAX. VALUE IS 59
00591A FD35 22 1B FD52          BHI    L03      TOO LARGE - CLEAR TIME VALUES
00592A FD37 BD FE18  A          JSR    CHKBCD   SET CARRY IF NOT A PROPER BCD VALUE
00593A FD3A 25 16 FD52          BCS    L03      NOT BCD - CLEAR TIME VALUES
00594                      *         CHECK MINUTES COUNT
00595A FD3C 96 0A    A          LDAA   MIN      GET MINUTES COUNTER
00596A FD3E 81 59    A          CMPA   #$59     MAX. VALUE IS 59 (BCD)
00597A FD40 22 10 FD52          BHI    L03      TOO LARGE - CLEAR TIME VALUES
00598A FD42 BD FE18  A          JSR    CHKBCD   SET CARRY IS NOT A BCD VALUE
00599A FD45 25 0B FD52          BCS    L03      NOT BCD - CLEAR TIME VALUES
00600                      *         CHECK HOURS
00601A FD47 96 09    A          LDAA   HR       GET HOURS
00602A FD49 81 23    A          CMPA   #$23     MAX. VALUE IS 23 (BCD)
00603A FD4B 22 05 FD52          BHI    L03      TOO LARGE - CLEAR TIME VALUES
00604A FD4D BD FE18  A          JSR    CHKBCD   SET CARRY IF NOT A BCD VALUE
00605A FD50 24 15 FD67          BCC    L04      OK --
00606                      *         TIME DATA IS IN ERROR - ZERO OUT ALL TIME VALUES
00607A FD52 7F 0030  A L03      CLR    PUCR     1/60TH SEC. COUNTER
00608A FD55 7F 000C  A          CLR    TSEC     1/10TH SEC. COUNTER
00609A FD58 7F 000B  A          CLR    SEC      SECOND COUNTER
00610A FD5B 7F 0031  A          CLR    SSEC     SIX SECOND COUNTER
00611A FD5E 7F 000A  A          CLR    MIN      MINUTES COUNTER
00612A FD61 7F 0032  A          CLR    SMIN     SIX MINUTE COUNTER
00613A FD64 7F 0009  A          CLR    HR       HOUR COUNTER
00614                      *         CHECK DAY OF WEEK (MUST BE <= 7)
00615A FD67 96 08    A L04      LDAA   DAOFWK   GET DAY OF WEEK
00616A FD69 81 07    A          CMPA   #7       CHECK IF VALUE OK
00617A FD6B 23 04 FD71          BLS    L05      YES -- VALUE IS OK
00618A FD6D 86 01    A          LDAA   #1       VALUE TOO LARGE - SET IT TO MONDAY
00619A FD6F 97 08    A          STAA   DAOFWK   SAVE IT
00620                      *
00621                      *         SAVE DATE AND TIME OF LAST POWER-DOWN
00622                      *
00623A FD71 96 03    A L05      LDAA   YEAR     GET YEAR
00624A FD73 97 87    A          STAA   LPYEAR   SAVE IT
00625A FD75 96 04    A          LDAA   MONTH    GET MONTH
00626A FD77 97 88    A          STAA   LPMNTH   SAVE IT
```

```
PAGE  005  POWRON  .SA:1  POWRON

00627A FD79 96 05     A        LDAA   DAOFMO   GET DAY OF MONTH
00628A FD7B 97 89     A        STAA   LFDAY    SAVE IT
00629A FD7D 96 09     A        LDAA   HR       GET HOUR
00630A FD7F 97 8A     A        STAA   LFHR     SAVE IT
00631A FD81 96 0A     A        LDAA   MIN      GET MINUTE
00632A FD83 97 8B     A        STAA   LFMIN    SAVE IT
00633                  *
00634                  *
00635                  *
00636                  *
00637                  *
00638                  *-----------------------------------------
00639A FD85 0E                 CLI             ENABLE INTERRUPTS
00640                  *-----------------------------------------
00641                  *
00642                  *
00643                  *
00644                  *
```

```
PAGE  006  POWRON  .SA:1  POWRON

00646                      *
00647                      *        INITIALIZE OTHER VARIABLES
00648                      *
00649A FD86 CE C060  A          LDX     #$C060   TURN ON LIGHTS 8 & 9, BLANK OUT DISPLAY
00650A FD89 FF 5008  A          STX     PORT8
00651A FD8C 86 FF    A          LDAA    #$FF     TURN ON LIGHTS 0->7
00652A FD8E B7 500A  A          STAA    PORTA
00653A FD91 86 09    A          LDAA    #9       KEYBOARD STATE = 9 (DISPLAY CHECKSUM RESULTS)
00654A FD93 97 20    A          STAA    KYSTAT   SAVE IT
00655A FD95 7F 002C  A          CLR     DISTMR   START DISPLAY TIMER (FOR STATE 9)
00656A FD98 CE 0000  A          LDX     #0
00657A FD9B DF 33    A          STX     TMFLG1   CLEAR TMFLG1,TMFLG2
```

```
PAGE  007  POWRON   .SA:1   POWRON

00659                       *
00660                       *       UPDATE TIME
00661                       *
00662A FD9D 7C 0036  A            INC     DWNCNT   BUMP TIMES-DOWN COUNT (POWER FAILURE COUNT)
00663A FDA0 96 37    A            LDAA    ALARMS   GET ALARM BYTE
00664A FDA2 8A 80    A            ORAA    #%10000000 SET POWER FAILURE BIT
00665                       *
00666                       *       CHECK FOR LONG POWER DOWN (>254 MIN. 59 SEC.)
00667                       *       THIS IS INDICATED BY 255 IN THE MINUTES DOWN-TIME ACCUMULATOR
00668                       *       AND 0 IN THE SECONDS DOWN-TIME ACCUMULATOR
00669                       *
00670A FDA4 F6 5000  A            LDAB    PORTO    GET DOWN TIME ACCUMULATOR (MINUTES)
00671A FDA7 C1 FF    A            CMPB    #255     DTA OVERRUN?
00672A FDA9 26 08 FDB3            BNE     L06      NO - UPDATE TIME
00673                       *       LONG POWER-DOWN - SET ERROR FLAGS
00674A FDAB D6 44    A            LDAB    ERCODE   GET ERROR CODE
00675A FDAD CA 04    A            ORAB    #%00000100 SET CLOCK ERROR BIT
00676A FDAF D7 44    A            STAB    ERCODE   SAVE IT
00677A FDB1 8A 02    A            ORAA    #%00000010 SET LONG POWER DOWN BIT
00678A FDB3 97 37    A L06        STAA    ALARMS   STORE BACK ALARMS
00679                       *
00680                       *       UPDATE SECONDS
00681                       *
00682A FDB5 F6 500F  A            LDAB    PORTF    GET SECONDS DOWN
00683A FDB8 96 0C    A            LDAA    TSEC     GET 1/10 SECOND COUNT
00684A FDBA 8B 05    A            ADDA    #5       ADD 5/10 SEC FOR AVERAGE DOWNTIME ACCUMULATOR LOSS
00685A FDBC 81 09    A L07        CMPA    #9       CHECK FOR
00686A FDBE 23 05 FDC5            BLS     L08      LESS THAN FULL SECOND
00687A FDC0 80 0A    A            SUBA    #10      FULL SECOND REACHED
00688A FDC2 5C                    INCB             INCREMENT SECONDS
00689A FDC3 20 F7 FDBC            BRA     L07      CHECK AGAIN
00690A FDC5 97 0C    A L08        STAA    TSEC     STORE BACK
00691                       *
00692                       *       1/10TH SEC. COUNTER UPDATED - NOW DO SECONDS
00693                       *       FULL SECONDS ADJUSTMENT NOW IN REGISTER B (HEX)
00694                       *
00695                       *       HERE, TEMP03 AND TEMP04 ARE USED TO STORE MINUTES ADJUSTMENT
00696                       *
00697A FDC7 7F 0080  A            CLR     TEMP03   CLEAR MSB OF ADJUSTMENT
00698A FDCA B6 5000  A            LDAA    PORTO    GET MINUTES DOWN ACCUMULATOR
00699A FDCD 97 81    A            STAA    TEMP04   SAVE IT
00700A FDCF 5D                    TSTB             SECONDS DOWN = 0?
00701A FDD0 27 17 FDE9            BEQ     UPDMIN   YES - UPDATE MINUTES
00702A FDD2 96 0B    A            LDAA    SEC      GET SECONDS (BEFORE POWER FAILURE)
00703                       *
00704A FDD4 8B 01    A L09        ADDA    #1       ADD 1 SECOND
00705A FDD6 19                    DAA              CONVERT TO BCD
00706A FDD7 81 60    A            CMPA    #$60     FULL MINUTE?
00707A FDD9 26 09 FDE4            BNE     L11      NOT YET
00708A FDDB 7C 0081  A            INC     TEMP04   INCREMENT MINUTES ADJUSTMENT (LSB)
00709A FDDE 26 03 FDE3            BNE     L10      NO OVERFLOW
00710A FDE0 7C 0080  A            INC     TEMP03   CATCH OVERFLOW IN MSB
00711A FDE3 4F         L10        CLRA             CLEAR SECONDS COUNT
00712A FDE4 5A         L11        DECB             DECREMENT SECONDS COUNTER (NOT BCD)
00713A FDE5 26 ED FDD4            BNE     L09      NOT FINISHED SECONDS UPDATE
00714A FDE7 97 0B    A            STAA    SEC      SAVE SECONDS
00715                       *
00716                       *       SECONDS NOW UPDATED - DO MINUTES
```

```
PAGE  008  POWRON  .SA:1  POWRON

00717                    *       NUMBER OF MINUTES TO UPDATE BY IS IN TEMP03 (MSB) AND TEMP04 (LSB)
00718                    *
00719A FDE9 DE 80      A UPDMIN LDX    TEMP03    GET MINUTES ADJUSTMENT
00720A FDEB 27 06 FDF3          BEQ    L99       NO MINUTES TO UPDATE
00721A FDED D6 34      A L12    LDAB   TMFLG2    GET LONG TIME FLAGS (SWI EXPECTS THIS)
00722                    *- - - - - - - - - - - - - - - - - - - -
00723A FDEF 3F                  SWI              SOFTWARE INTERRUPT
00724                    *- - - - - - - - - - - - - - - - - - - -
00725A FDF0 09                  DEX              DECREMENT MINUTES ADJUSTMENT
00726A FDF1 26 FA FDED          BNE    L12       DO IT AGAIN
00727                    *
00728A FDF3 7F 5000    A L99    CLR    PORT0     CLEAR DTA
```

```
00730                     *        DO PROM CHECKSUM CALCULATIONS AND PREVENT A PARALLEL CALCULATION
00731                     *        BEING DONE BY THE CHEK60 ROUTINE
00732                     *
00733A FDF6 96 6C    A         LDAA   MFLAG     GET BYTE CONTAINING CHECKSUM FLAG
00734A FDF8 84 DF    A         ANDA   #%11011111 RESET BIT 5, IE, DO NOT DO ANOTHER PROM CHECKSUM TODAY
00735A FDFA 97 6C    A         STAA   MFLAG     SAVE FLAG
00736A FDFC CE C000  A         LDX    #$C000    BEGINNING OF PROM
00737A FDFF DF 84    A         STX    PADDR     STORE INTO PROM CHECKSUM ADDRESS SAVE AREA
00738A FE01 CE 0000  A         LDX    #0
00739A FE04 DF 82    A         STX    PCHECK    CLEAR PROM CHECKSUM
00740A FE06 7F 0086  A         CLR    PRERFL    CLEAR PROM CHECKSUM ERRORS
00741A FE09 BD FF30  A         JSR    CALPRO    CALCULATE PROM CHECKSUM
00742                     *
00743                     *        DO RAM TEST
00744                     *
00745A FE0C 96 6C    A         LDAA   MFLAG     GET FLAGS
00746A FE0E 84 BF    A         ANDA   #%10111111 INDICATE THIS IS A POWER UP RAM TEST
00747A FE10 97 6C    A         STAA   MFLAG     SAVE IT
00748A FE12 BD F760  A         JSR    RAMCHK    PERFORM RAM TEST
00749                     *
00750                     *
00751                     *        END OF POWRON ROUTINE, INVOKE EXECUTIVE ROUTINE
00752                     *
00753                     *
00754A FE15 7E FE30  A         JMP    EXEC
```

```
PAGE  010  POWRON  .SA:1  POWRON

00756                  ******************************************************************
00757                  *         SUBROUTINE TO CHECK IF THE A REGISTER CONTAINS
00758                  *         A VALID BCD NUMBER
00759                  *.         IF THE NUMBER IS A VALID BCD VALUE, THE CARRY FLAG
00760                 .*          (IE, C BIT) IS CLEARED
00761                  *          OTHERWISE, THE CARRY BIT IS SET
00762                  *         THIS SUBROUTINE MIGHT DESTROY THE A REGISTER
00763                  ******************************************************************
00764A FE18 81 9F    A CHKBCD CMPA   #$9F      BCD VALUES MUST BE LESS THAN 9F
00765A FE1A 22 08 FE24        BHI    BADOUT    TOO BIG - NOT A BCD VALUE
00766A FE1C 84 0F    A        ANDA   #$0F      EXAMINE LOWER NIBBLE
00767A FE1E 81 09    A        CMPA   #9        LOWER NIBBLE MUST NOT EXCEED 9
00768A FE20 22 02 FE24        BHI    BADOUT    THIS ONE DOES - NOT BCD
00769A FE22 0C                CLC              INDICATE VALID BCD VALUE
00770A FE23 39                RTS
00771A FE24 0D         BADOUT SEC              INDICATE BAD BCD VALUE
00772A FE25 39                RTS
00773                         END
TOTAL ERRORS 00000--00000
```

```
PAGE  001  EQUATE  .SA:1  EXEC

       OPT    L
       NAM    EXEC
*
*
*         EXECUTIVE ROUTINE
*
*
*             ****************************************************
*             *                                                  *
*             *   PROGRAMMER:   W.J. WIERCIENSKI                 *
*             *                                                  *
*             *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS  *
*             *   TRANSPORTATION TECHNOLOGY AND ENERGY           *
*             *   1201 WILSON AVENUE                             *
*             *   DOWNSVIEW, ONTARIO  M3M 1J8                    *
*             *                                                  *
*             *   TELEPHONE (416) 248-7141                       *
*             *                                                  *
*             *   NOVEMBER  1982 AD                              *
*             *                                                  *
*             ****************************************************
*
*
*         THIS PROGRAM CALLS ALL PROGRAMS WHICH ARE NOT
*         RUN DURING THE INTERRUPT PERIODS
*
*
```

```
PAGE  002  EXEC    .SA:1  EXEC

00526A FE30                    ORG    EXEC
00527                 *
00528                 *           WAIT IF THERE IS NOTHING TO DO - PROCEED OTHERWISE
00529                 *
00530A FE30 96 33    A         LDAA   TMFLG1   SHORT PERIOD TIME FLAGS
00531A FE32 26 03 FE37         BNE    GO       IF ANY SET GO ON
00532A FE34 3E                 WAI             OTHERWISE WAIT FOR INTERRUPT
00533A FE35 20 F9 FE30         BRA    EXEC     & GO CHECK FLAGS
00534                 *
00535                 *           EXECUTE SHORT PERIOD PROGRAMS
00536                 *
00537A FE37 CE FE76  A GO      LDX    #EXTABL  SET X REG FOR SHORT TIME PERIOD TABLE
00538A FE3A DF 38    A         STX    EXINDX
00539A FE3C 7F 0033  A         CLR    TMFLG1   RESET TIME FLAGS
00540A FE3F 8D 10 FE51         BSR    FIND     EXECUTE ANY NEEDED ROUTINES
00541                 *
00542                 *           EXECUTE LONG PERIOD PROGRAMS
00543                 *
00544A FE41 96 34    A         LDAA   TMFLG2   LONG TERM FLAGS
00545A FE43 27 EB FE30         BEQ    EXEC     NONE SET, GO CHECK SHORT TERM FLAGS
00546A FE45 CE FE86  A         LDX    #EXTABL+16 SET X REG FOR LONG TIME FLAGS
00547A FE48 DF 38    A         STX    EXINDX
00548A FE4A 7F 0034  A         CLR    TMFLG2   RESET TIME FLAGS
00549A FE4D 8D 02 FE51         BSR    FIND     EXECUTE NEEDED ROUTINES
00550A FE4F 20 DF FE30         BRA    EXEC     BACK FOR MORE WORK
```

```
PAGE  003  EXEC    .SA:1  EXEC

00552                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00553                        *
00554                        *       SUBROUTINE TO EXECUTE THE PROGRAMS ASSOCIATED WITH
00555                        *       THE BITS SET IN THE TIME FLAGS PASSED IN REG. A -
00556                        *       ON INPUT, REG. X AND EXINDX BOTH MUST BE SET TO THE
00557                        *       BEGINNING OF THE APPROPRIATE TABLE FOR THESE TIME
00558                        *       FLAGS
00559                        *
00560                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
00561                        *
00562A FE51 44               FIND    LSRA            PUT TIME BIT INTO CARRY BIT
00563A FE52 97 35      A             STAA   EXFLG    SAVE SHIFTED TIME FLAGS
00564A FE54 24 15 FE6B               BCC    NXFLG    BIT NOT SET - TRY NEXT ONE
00565A FE56 EE 00      A             LDX    0,X      GET SUBROUTINE TABLE ADDRESS
00566A FE58 DF 3A      A F3          STX    SBRINX   SAVE SUBR TABLE ADDRESS
00567A FE5A EE 00      A             LDX    0,X      GET SUBR TABLE MEMBER
00568A FE5C 8C FFFF    A             CPX    #$0FFFF  END OF TABLE FOR THIS FLAG?
00569A FE5F 27 08 FE69               BEQ    EOTBL    YES --
00570A FE61 AD 00      A             JSR    0,X      NO -- EXECUTE THE SUBROUTINE
00571A FE63 DE 3A      A             LDX    SBRINX   GET SUBR TABLE ADDRESS BACK
00572A FE65 08                       INX             MOVE TO NEXT TABLE ENTRY
00573A FE66 08                       INX
00574A FE67 20 EF FE58               BRA    F3       CHECK OUT NEXT TABLE ENTRY
00575A FE69 96 35      A EOTBL       LDAA   EXFLG    GET BACK TIME FLAG POINTER (SHIFTED)
00576A FE6B 27 08 FE75 NXFLG         BEQ    F4       - NO MORE FLAGS SET
00577A FE6D DE 38      A             LDX    EXINDX   GET TIME PERIOD TABLE INDEX
00578A FE6F 08                       INX             MOVE TO NEXT TIME PERIOD ENTRY
00579A FE70 08                       INX
00580A FE71 DF 38      A             STX    EXINDX   SAVE THE LOCATION
00581A FE73 20 DC FE51               BRA    FIND     PROCESS NEXT TIME FLAG BIT
00582A FE75 39               F4      RTS
00583                        *
00584                        *       POINTERS TO LISTS OF SUBROUTINES TO EXECUTE AT
00585                        *       VARIOUS TIME FRAMES
00586                        *
00587A FE76      FE96  A EXTABL FDB    HZ60TB
00588A FE78      FEA6  A        FDB    HZ30TB
00589A FE7A      FEAA  A        FDB    HZ20TB
00590A FE7C      FEAE  A        FDB    HZ10TB
00591A FE7E      FEB6  A        FDB    HZ2TB
00592A FE80      FEBC  A        FDB    SC1TB
00593A FE82      FEC0  A        FDB    SC6TB
00594A FE84      FECE  A        FDB    SC30TB
00595A FE86      FED2  A        FDB    MN1TB
00596A FE88      FED6  A        FDB    MN6TB
00597A FE8A      FEDA  A        FDB    MN15TB
00598A FE8C      FEDE  A        FDB    MN20TB
00599A FE8E      FEE2  A        FDB    MN30TB
00600A FE90      FEE6  A        FDB    HR1TB
00601A FE92      FEEC  A        FDB    HR24TB
00602A FE94      FEF2  A        FDB    SP24TB
```

```
PAGE  004  EXEC    .SA:1  EXEC

00604                   *
00605                   *          TABLES OF SUBROUTINES TO EXECUTE IN EACH TIME
00606                   *             PERIOD.  TABLE ENTRIES ARE STARTING ADDRESSES
00607                   *             OF ROUTINES TO EXECUTE IN ORDER OF EXECUTION.
00608                   *             EACH TABLE MUST TERMINATE WITH AN END-OF-TABLE
00609                   *             MARKER ($0FFFF)
00610                   *
00611A FE96    FF20   A HZ60TB FDB    CHEK60
00612A FE98    F750   A        FDB    RAM60
00613A FE9A    C350   A        FDB    PCS2
00614A FE9C    D4A0   A        FDB    PCS3
00615A FE9E    D550   A        FDB    PCS3A
00616A FEA0    CBB0   A        FDB    PCS3B
00617A FEA2    CC00   A        FDB    PCS3C
00618A FEA4    FFFF   A        FDB    $0FFFF
00619A FEA6    F800   A HZ30TB FDB    KEYBD
00620A FEA8    FFFF   A        FDB    $0FFFF
00621A FEAA    FFFF   A HZ20TB FDB    $0FFFF
00622A FEAC    FFFF   A        FDB    $0FFFF
00623A FEAE    F670   A HZ10TB FDB    TIMER1
00624A FEB0    CC50   A        FDB    PCS8
00625A FEB2    CDA0   A        FDB    PCS8A
00626A FEB4    FFFF   A        FDB    $0FFFF
00627A FEB6    FFFF   A HZ2TB  FDB    $0FFFF
00628A FEB8    FFFF   A        FDB    $0FFFF
00629A FEBA    FFFF   A        FDB    $0FFFF
00630A FEBC    F6F0   A SC1TB  FDB    MAGINT
00631A FEBE    FFFF   A        FDB    $0FFFF
00632A FEC0    C400   A SC6TB  FDB    PCS4
00633A FEC2    C840   A        FDB    PCS4A
00634A FEC4    C550   A        FDB    PCS6
00635A FEC6    C950   A        FDB    PCS6A
00636A FEC8    C4A0   A        FDB    PCS5
00637A FECA    C8D0   A        FDB    PCS5A
00638A FECC    FFFF   A        FDB    $0FFFF
00639A FECE    FFFF   A SC30TB FDB    $0FFFF
00640A FED0    FFFF   A        FDB    $0FFFF
00641A FED2    F6C0   A MN1TB  FDB    TIMER2
00642A FED4    FFFF   A        FDB    $0FFFF
00643A FED6    FFFF   A MN6TB  FDB    $0FFFF
00644A FED8    FFFF   A        FDB    $0FFFF
00645A FEDA    FFFF   A MN15TB FDB    $0FFFF
00646A FEDC    FFFF   A        FDB    $0FFFF
00647A FEDE    FFFF   A MN20TB FDB    $0FFFF
00648A FEE0    FFFF   A        FDB    $0FFFF
00649A FEE2    FFFF   A MN30TB FDB    $0FFFF
00650A FEE4    FFFF   A        FDB    $0FFFF
00651A FEE6    C6A0   A HR1TB  FDB    PCS7
00652A FEE8    CF00   A        FDB    PCS7A
00653A FEEA    FFFF   A        FDB    $0FFFF
00654A FEEC    FF00   A HR24TB FDB    CHEK24
00655A FEEE    D250   A        FDB    HEADER
00656A FEF0    FFFF   A        FDB    $0FFFF
00657A FEF2    FFFF   A SP24TB FDB    $0FFFF
00658A FEF4    FFFF   A        FDB    $0FFFF
00659                          END
TOTAL ERRORS 00000--00000
```

```
       OPT    L
       NAM    CALPRO
*
*
*         CALCULATE PROM CHECKSUM
*
*
*             ****************************************************
*             *                                                  *
*             *   PROGRAMMER:   W.J. WIERCIENSKI                 *
*             *                                                  *
*             *   MINISTRY OF TRANSPORTATION AND COMMUNICATIONS  *
*             *   TRANSPORTATION TECHNOLOGY AND ENERGY           *
*             *   1201 WILSON AVENUE                             *
*             *   DOWNSVIEW, ONTARIO  M3M 1J8                    *
*             *                                                  *
*             *   TELEPHONE (416) 248-7141                       *
*             *                                                  *
*             *   NOVEMBER  1982 AD                              *
*             *                                                  *
*             ****************************************************
*
*
*          THE FOLLOWING CODING IS USED FOR SYSTEM FUNCTIONS AND IN THE CALCULATION
*          AND CHECKING OF THE PROM CHECKSUMS. IF ANY OF THE 2K INCREMENTS IS
*          FAULTY, THE APPROPRIATE FLAG IS SET IN THE PROM ERROR FLAG BYTE PRERFL
*
```

```
PAGE  002  CALPRO  .SA:1  CALPRO

00526A FF00                     ORG    CHEK24
00527                     *
00528                     *         24 HR ROUTINE - INITIALIZES CONTROLLER FOR THE CALCULATION OF THE PROM CHECKSUM
00529                     *
00530A FF00 96 6C    A          LDAA   MFLAG    GET BYTE CONTAINING PROM CHECKSUM FLAG
00531A FF02 8A 20    A          ORAA   #%00100000 SET BIT 5  IE. DO PROM CHECKSUM
00532A FF04 97 6C    A          STAA   MFLAG    SAVE IT
00533A FF06 CE C000  A          LDX    #$C000   BEGINNING OF PROM
00534A FF09 DF 84    A          STX    PADDR    STORE INTO PROM CHECKSUM SAVE AREA
00535A FF0B CE 0000  A          LDX    #0
00536A FF0E DF 82    A          STX    PCHECK   CLEAR PROM CHECKSUM
00537A FF10 7F 0086  A          CLR    PRERFL   CLEAR PROM CHECKSUM ERRORS
00538A FF13 39                  RTS
```

```
PAGE  003  CALPRO  .SA:1  CALPRO

00540                     *
00541                     *       THIS SUBROUTINE PERFORMS THE PROM CHECKSUM DAILY STARTING RIGHT AFTER
00542                     *       MIDNIGHT. IT CONTINUES 1 BYTE AT A TIME EVERY 1/60TH OF A SECOND
00543                     *       UNTIL THE JOB IS DONE.
00544                     *
00545A FF20                     ORG    CHEK60
00546A FF20 96 6C    A          LDAA   MFLAG     GET BYTE CONTAINING PROM CHECKSUM FLAG
00547A FF22 85 20    A          BITA   #%00100000 CHECK IF CHECKSUM CALCULATIONS ALREADY FINISHED
00548A FF24 27 03 FF29          BEQ    L1        FLAG NOT SET - NOTHING TO BE DONE
00549A FF26 BD FF30  A          JSR    CALPRO    GO TO CALCULATE PROM CHECKSUM ON ONE BYTE
00550A FF29 39           L1     RTS              EXIT
```

```
PAGE  004  CALPRO  .SA:1  CALPRO

00552                        *
00553                        *      CALCULATE PROM CHECKSUM
00554                        *
00555                        *      THE CHECKSUM IS CALCULATED FOR EVERY 2K OF PROM MEMORY. IT IS THE 16 LEAST
00556                        *      SIGNIFICANT BITS OF THE SUM OF ALL DATA BYTES, REPRESENTED IN HEX.
00557                        *      THE CALCULATED CHECKSUM IS COMPARED WITH THE VALUE STORED FOR EACH
00558                        *      INDIVIDUAL 2K OF PROM. THERE IS A PROVISION FOR EIGHT CHECKSUMS AT THE LOCATION
00559                        *      PCHKSM IN THE TOP MOST CHIP. THESE VALUES ARE OBTAINED FROM THE DATA I/O PROM
00560                        *      PROGRAMMER WHEN THE CHIPS ARE PROGRAMMED.
00561                        *
00562                        *      THIS PROCEDURE CALCULATES THE CHECKSUMS FOR THE ENTIRE PROM MEMORY WHEN
00563                        *      CALLED UPON BY THE POWRON SUBROUTINE. OTHERWISE IT CALCULATES 1 BYTE EVERY 1/60TH SEC.
00564                        *
00565A FF30                       ORG    CALPRO
00566                        *
00567                        *      UPDATE PROM CHECKSUM BY 1 BYTE
00568                        *
00569A FF30 DE 84    A            LDX    PADDR    GET ADDRESS OF BYTE TO BE ADDED TO THE CHECKSUM NEXT
00570A FF32 96 82    A C1         LDAA   PCHECK   MSB OF PROM CHECKSUM
00571A FF34 D6 83    A            LDAB   PCHECK+1 LSB OF PROM CHECKSUM
00572A FF36 EB 00    A            ADDB   0,X      ADD VALUE OF NEXT BYTE
00573A FF38 89 00    A            ADCA   #0       ADD CARRY IF SET
00574A FF3A 97 82    A            STAA   PCHECK   (SAVE
00575A FF3C D7 83    A            STAB   PCHECK+1 CHECKSUM)
00576                        *
00577                        *       CHECK FOR END OF 2K
00578                        *
00579A FF3E D6 85    A            LDAB   PADDR+1  GET LSB OF ADDRESS
00580A FF40 C1 FF    A            CMPB   #$FF     END OF A PAGE (256 BYTES)?
00581A FF42 26 02 FF46            BNE    C80      DEFINITELY NOT END OF 2K
00582A FF44 20 03 FF49            BRA    C1X      CONTINUE
00583A FF46 7E FFC5  A C80        JMP    C8       ONTO NEXT BYTE
00584A FF49 96 84    A C1X        LDAA   PADDR    END OF 2K POSSIBLE - GET MSB OF ADDRESS
00585A FF4B 84 07    A            ANDA   #%00000111 GET RID OF BITS 3 TO 7
00586A FF4D 81 07    A            CMPA   #%00000111 CHECK FOR 7 OR F
00587A FF4F 26 74 FFC5            BNE    C8       NOT THE END OF 2K
00588                        *
00589                        *       END OF 2K OF PROM - COMPARE CHECKSUMS
00590                        *
00591A FF51 4F                    CLRA            FOR FIRST 8K OF PROM, DISPLACEMENT INTO CHECKSUM TABLE = 0
00592A FF52 D6 84    A            LDAB   PADDR    GET MSB OF ADDRESS
00593A FF54 C1 E0    A            CMPB   #$E0     FIND IF IT IS FIRST OR SECOND 8K
00594A FF56 23 02 FF5A            BLS    C2       BRANCH IF C OR D (FIRST 8K OF PROM)
00595                        *
00596                        *       CALCULATE ADDRESS OF PROPER CHECKSUM
00597                        *
00598A FF58 8B 08    A            ADDA   #8       DISPLACEMENT INTO PROM CHECKSUM LIST (SECOND 8K OF PROM)
00599A FF5A C4 18    A C2         ANDB   #%00011000 KEEP BITS 3 AND 4
00600A FF5C 54                    LSRB
00601A FF5D 54                    LSRB            CALCULATE POSITION
00602A FF5E 1B                    ABA             NOW REG. A CONTAINS DISPLACEMENT FROM BEGINNING OF CHECKSUM TABLE
00603A FF5F 16                    TAB             SAVE FOR FUTURE USE (SETTING OF PROM ERROR FLAG)
00604                        *
00605                        *       DETERMINE THE ACTUAL LOCATION OF THE PROM CHECKSUM
00606                        *
00607A FF60 CE FFD6  A            LDX    #PCHKSM  BEGINNING OF CHECKSUM LIST
00608A FF63 DF 80    A            STX    TEMP03   SAVE
00609A FF65 96 81    A            LDAA   TEMP03+1 LSB OF PROM CHECKSUM LIST ADDRESS
```

```
00610A FF67 1B                  ABA              ADD IN DISPLACEMENT
00611A FF68 97 81     A         STAA   TEMP03+1 STORE BACK
00612A FF6A 96 80     A         LDAA   TEMP03   GET MSB OF ADDRESS
00613A FF6C 89 00     A         ADCA   #0       ADD IN CARRY
00614A FF6E 97 80     A         STAA   TEMP03   STORE BACK
00615A FF70 DE 80     A         LDX    TEMP03   LOCATION OF CHECKSUM
00616                      *
00617                      *        COMPARE THE CALCULATED CHECKSUM WITH THE GIVEN CHECKSUM
00618                      *
00619A FF72 96 83     A         LDAA   PCHECK+1 LSB OF CALCULATED CHECKSUM
00620A FF74 A1 01     A         CMPA   1,X      COMPARE WITH EXPECTED LSB
00621A FF76 26 06 FF7E          BNE    C3       NOT EQUAL -> CHECKSUMS DO NOT MATCH
00622A FF78 96 82     A         LDAA   PCHECK   MSB OF CALCULATED CHECKSUM
00623A FF7A A1 00     A         CMPA   0,X      COMPARE WITH EXPECTED MSB
00624A FF7C 27 14 FF92          BEQ    C6       EQUAL -> CHECK IF ALL CHIPS COMPLETED
00625                      *
00626                      *        THE PROM CHECKSUMS DO NOT MATCH - SET THE APPROPRIATE BIT
00627                      *
00628A FF7E 17             C3   TBA              GET DISPLACEMENT INTO CHECKSUM TABLE
00629A FF7F C6 01     A         LDAB   #%00000001 GET A BYTE READY FOR SETTING THE PROM CHECKSUM ERROR BIT
00630A FF81 44                  LSRA             SHIFT RIGHT TO DIVIDE BY 2
00631A FF82 81 00     A C4      CMPA   #0       SEE IF THE CORRECT BIT HAS BEEN FOUND
00632A FF84 27 04 FF8A          BEQ    C5       GO TO SET THE BIT
00633A FF86 58                  ASLB             GET THE SETTING BYTE READY FOR THE NEXT POSSIBLE ERROR BIT
00634A FF87 4A                  DECA             CONTINUE THE COUNT DOWN
00635A FF88 20 F8 FF82          BRA    C4       GO TRY AGAIN
00636A FF8A 96 86     A C5      LDAA   PRERFL   GET PROM ERROR FLAG BYTE
00637A FF8C D7 82     A         STAB   PCHECK+0 SAVE THE SETTING BYTE
00638A FF8E 9A 82     A         ORAA   PCHECK+0 SET THE APPROPRIATE PROM ERROR FLAG BIT
00639A FF90 97 86     A         STAA   PRERFL   SAVE UPDATED PROM ERROR FLAGS
00640                      *
00641                      *        CHECK FOR END OF PROM MEMORY
00642                      *
00643A FF92 DE 84     A C6      LDX    PADDR    GET ADDRESS
00644A FF94 8C FFFF   A         CPX    #$FFFF   IS THIS THE END OF PROM
00645A FF97 26 27 FFC0          BNE    C7       NOT THE END OF PROM
00646A FF99 96 6C     A         LDAA   MFLAG    GET FLAGS
00647A FF9B 84 DF     A         ANDA   #%11011111 RESET FLAG
00648A FF9D 97 6C     A         STAA   MFLAG    SAVE RESULT
00649                      *
00650                      *        TAKE CARE OF PROM CHECKSUM ERROR FLAG IN ERCODE
00651                      *
00652A FF9F D6 44     A         LDAB   ERCODE   GET ERROR CODE BYTE
00653A FFA1 96 86     A         LDAA   PRERFL   CHECK IF THERE IS ANY BAD PROM MEMORY
00654A FFA3 26 04 FFA9          BNE    C61      NO
00655A FFA5 C4 FD     A         ANDB   #%11111101 NO PROBLEMS - RESET ERROR FLAG
00656A FFA7 20 02 FFAB          BRA    C62
00657A FFA9 CA 02     A C61     ORAB   #%00000010 SET ERROR CODE
00658A FFAB D7 44     A C62     STAB   ERCODE   SAVE ERROR CODE BYTE
00659                      *
00660                      *        INITIATE RAM TEST
00661                      *
00662A FFAD CE 0000   A         LDX    #0       SET RAM TEST STARTING ADDRESS
00663A FFB0 DF 84     A         STX    PADDR    SAVE IT
00664A FFB2 DF 2E     A         STX    RAMERR   CLEAR RAM ERROR FLAGS
00665A FFB4 CE 0100   A         LDX    #$0100
00666A FFB7 DF 82     A         STX    PCHECK   SET RAM CHIP BIT INDICATOR
00667A FFB9 96 6C     A         LDAA   MFLAG    GET MASTER FLAGS
```

```
PAGE  006  CALPRO  .SA:1  CALPRO

00668A FFBB 8A 40    A         ORAA   #%01000000 START RAM TEST
00669A FFBD 97 6C    A         STAA   MFLAG     SAVE FLAGS
00670A FFBF 39                 RTS              EXIT
00671                     *
00672                     *         NOT THE END OF PROM
00673                     *
00674A FFC0 CE 0000  A C7      LDX    #0        (CLEAR
00675A FFC3 DF 82    A         STX    PCHECK    CHECKSUM)
00676A FFC5 DE 84    A C8      LDX    PADDR     GET ADDRESS
00677A FFC7 08                 INX              INCREMENT TO THE NEXT BYTE
00678A FFC8 DF 84    A         STX    PADDR     SAVE ADDRESS
00679A FFCA 96 6C    A         LDAA   MFLAG     GET FLAGS
00680A FFCC 85 20    A         BITA   #%00100000 IS THE FLAG SET?
00681A FFCE 27 02 FFD2         BEQ    C9        CONTINUOUS CALCULATIONS ON POWER UP
00682A FFD0 20 03 FFD5         BRA    C10
00683A FFD2 7E FF32  A C9      JMP    C1        DO NEXT BYTE
00684A FFD5 39         C10     RTS              EXIT
00685                     *
00686                     *         TABLE OF CHECKSUMS
00687                     *
00688A FFD6     F800 A PCHKSM FDB     $F800     CHECKSUM FOR C000 TO C7FF
00689A FFD8     F800 A         FDB    $F800     CHECKSUM FOR C800 TO CFFF
00690A FFDA     F800 A         FDB    $F800     CHECKSUM FOR D000 TO D7FF
00691A FFDC     F800 A         FDB    $F800     CHECKSUM FOR D800 TO DFFF
00692A FFDE     F800 A         FDB    $F800     CHECKSUM FOR E000 TO E7FF
00693A FFE0     F800 A         FDB    $F800     CHECKSUM FOR E800 TO EFFF
00694A FFE2     F800 A         FDB    $F800     CHECKSUM FOR F000 TO F7FF
00695A FFE4     F800 A         FDB    $F800     CHECKSUM FOR F800 TO FFFF
00696A FFE6     0000 A         FDB    0         COMPENSATING BYTES FOR CHECKSUM FOR LAST CHIP
00697                          END
TOTAL ERRORS 00000--00000
```

```
PAGE  001  EQUATE  .SA:1  VECTOR

00498                            OPT    L
00499                            NAM    VECTOR
00500                       *
00501                       *
00502                       *      INTERRUPT VECTORS
00503                       *
00504                       *
00505                       *          **************************************************
00506                       *          *                                                *
00507                       *          *  PROGRAMMER:   W.J. WIERCIENSKI                *
00508                       *          *                                                *
00509                       *          *  MINISTRY OF TRANSPORTATION AND COMMUNICATIONS *
00510                       *          *  TRANSPORTATION TECHNOLOGY AND ENERGY          *
00511                       *          *  1201 WILSON AVENUE                            *
00512                       *          *  DOWNSVIEW, ONTARIO  M3M 1J8                   *
00513                       *          *                                                *
00514                       *          *  TELEPHONE (416) 248-7141                      *
00515                       *          *                                                *
00516                       *          *  NOVEMBER  1982 AD                             *
00517                       *          *                                                *
00518                       *          **************************************************
00519                       *
00520                       *
00521                       *      THIS PROGRAM DIRECTS PROGRAM EXECUTION IN THE EVENT OF ANY OF THE
00522                       *      DIFFERENT INTERRUPTS TO THE REQUIRED ROUTINE TO PROPERLY SERVICE
00523                       *      THE RESPECTIVE INTERRUPT
00524                       *
00525                       *
00526                       *
00527A FFF6                      ORG    IGNORE
00528A FFF6 20 FE FFF6           BRA    IGNORE
00529                       *
00530A FFF8                      ORG    VECTOR
00531A FFF8     FB00  A          FDB    ITEST     HWI VECTOR
00532A FFFA     FB96  A          FDB    SWI       SWI VECTOR
00533A FFFC     FFF6  A          FDB    IGNORE    NMI VECTOR
00534A FFFE     FCD0  A          FDB    POWRON    RE-START VECTOR
00535                            END
TOTAL ERRORS 00000--00000
```